UM10375

LPC1311/13/42/43 User manual

Rev. 5 — 21 June 2012 User manual

Document information

Info	Content
Keywords	ARM Cortex-M3, microcontroller, USB, LPC1311, LPC1313, LPC1342, LPC1343, LPC1311/01, LPC1313/01
Abstract	LPC1311/13/42/43 user manual

Revision history

Rev	Date	Description
5	20120621	LPC1311/13/42/43 user manual
		Modifications: • Description of the IP_xxx bits in the interrupt priority registers updated (Table 78 to Table 92). • Description of the FUNC bits in register IOCON_PIO2_11 updated (Table 123). • ISP Go command description updated (ARM mode not allowed). See Table 325. • Description of interrupt use with IAP calls updated (Section 21.8.7). • Figure 4 updated ($\overline{\mathrm{RESET}}$ updated by internal reset). • Frequency values for FREQSEL bits in the WDTOSCCTRL register corrected (see Table 15). • SYSRESSTAT register access changed to R/W (Table 7). • SRAM use by the bootloader explained in Section 21.2. • Figure 9 “Standard I/O pin configuration” updated.
4	20110928	LPC1311/13/42/43 user manual
		Modifications: • PDSLEEPCFG register settings updated for parts LPC1311/01 and LPC1313/01 (see Section 3.5.45. • Figure 6 updated and power profile entry address updated in Section 5.4. • Figure 19 updated. • Description of UART modem interrupt added in Section 12.6.5.

Revision history *…continued*

Rev	Date	Description
3	20110614	LPC1311/13/42/43 user manual
		Modifications: • Parts LPC1311/01 and LPC1313/01 added. • Modifications to the user manual applicable to parts LPC1311/01 and LPC1313/01 only: – SSP1 added for part LPC1313FBD48/01 in Chapter 3 “LPC13xx System configuration” and Chapter 14 “LPC13xx SSP0/1”. – UART functions for part LPC1313FBD48/01 added in Table 128, Table 129, , and Table 138. – Use of IRC for entering deep power-down updated in Section 3.9.4.2. – Enable sequence for UART clock updated in Section 12.1. – Chapter 5 “LPC13xx Power profiles” added. – Register IOCON_DSR_LOC (Table 140), IOCON_DCD_LOC (Table 141), IOCON_RI_LOC (Table 142) added. – Programmable bit OD for pseudo open-drain mode added to IOCON registers in Chapter 7. – Chapter 19 “LPC13xx Windowed WatchDog Timer (WWDT)” added. • Editorial and formatting updates throughout the user manual. • Pull-up level for internal pull-ups specified in Section 7.3.2 and Section 8.4.1 and Section 8.4.2. • Description WDEN bit updated in Table 290 and Table 296 (WDMOD registers). • Section 3.7 “Start-up behavior” added. • NVIC priority register bit description updated in Section 6.6. • Description of GPIO data register updated in Section 9.4.1. • LPC1342FBD48 package added.
2	20100707	LPC1311/13/42/43 user manual
1	20091106	LPC1311/13/42/43 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**

UM10375

Chapter 1: LPC13xx Introductory information

Rev. 5 — 21 June 2012 User manual

1.1 Introduction

The LPC13xx are ARM Cortex-M3 based microcontrollers for embedded applications featuring a high level of integration and low power consumption. The ARM Cortex-M3 is a next generation core that offers system enhancements such as enhanced debug features and a higher level of support block integration.

The LPC13xx operate at CPU frequencies of up to 72 MHz. The ARM Cortex-M3 CPU incorporates a 3-stage pipeline and uses a Harvard architecture with separate local instruction and data buses as well as a third bus for peripherals. The ARM Cortex-M3 CPU also includes an internal prefetch unit that supports speculative branching.

The peripheral complement of the LPC13xx series includes up to 32 kB of flash memory, up to 8 kB of data memory, USB Device, one Fast-mode Plus (FM+) I^2C interface, one UART, four general purpose timers, and up to 42 general purpose I/O pins.

1.2 How to read this manual

This user manual describes parts LPC1311, LPC1313, LPC1342, LPC1343. Part-specific features and registers are listed at the beginning of each chapter.

Remark: The LPC13xx series consists of the LPC1300 series (parts LPC1311/13/42/43) and the LPC1300L series (parts LPC1311/01 and LPC1313/01). The LPC1300L series features the following enhancements over the LPC1300 series:

- Power profiles with lower power consumption in Active and Sleep modes.
- Four levels for BOD forced reset.
- Second SSP controller (LPC1313FBD48/01 only).
- Windowed Watchdog Timer (WWDT).
- Internal pull-up resistors pull up pins to full V_{DD} level.
- Programmable pseudo open-drain mode for GPIO pins.

1.3 Features

- ARM Cortex-M3 processor, running at frequencies of up to 72 MHz.
- ARM Cortex-M3 built-in Nested Vectored Interrupt Controller (NVIC).
- 32 kB (LPC1343/13)/16 kB (LPC1342)/8 kB (LPC1311) on-chip flash programming memory.
- 8 kB (LPC1343/13)/4 kB (LPC1342/11) SRAM.
- In-System Programming (ISP) and In-Application Programming (IAP) via on-chip bootloader software.
- Selectable boot-up: UART or USB (USB on LPC134x only).
- On LPC134x: USB MSC and HID on-chip drivers.
- Serial interfaces:

  - USB 2.0 full-speed device controller with on-chip PHY for device (LPC1342/43 only).
  - UART with fractional baud rate generation, modem, internal FIFO, and RS-485/EIA-485 support.
  - SSP controller with FIFO and multi-protocol capabilities.
  - Additional SSP controller on LPC1313FBD48/01.
  - I^{2}C-bus interface supporting full I^{2}C-bus specification and Fast-mode Plus with a data rate of 1 Mbit/s with multiple address recognition and monitor mode.
- Other peripherals:
  - Up to 42 General Purpose I/O (GPIO) pins with configurable pull-up/pull-down resistors.
  - Four general purpose counter/timers with a total of four capture inputs and 13 match outputs.
  - Programmable WatchDog Timer (WDT).
  - Programmable Windowed Watchdog Timer (WWDT) on LPC1311/01 and LPC1313/01.
  - System tick timer.
- Serial Wire Debug and Serial Wire Trace port.
- High-current output driver (20 mA) on one pin.
- High-current sink drivers (20 mA) on two I^{2}C-bus pins in Fast-mode Plus.
- Integrated PMU (Power Management Unit) to minimize power consumption during Sleep, Deep-sleep, and Deep power-down modes.
- Power profiles residing in boot ROM allowing to optimize performance and minimize power consumption for any given application through one simple function call. (LPC1300L series, on LPC1311/01 and LPC1313/01 only.)
- Three reduced power modes: Sleep, Deep-sleep, and Deep power-down.
- Single power supply (2.0 V to 3.6 V).
- 10-bit ADC with input multiplexing among 8 pins.
- GPIO pins can be used as edge and level sensitive interrupt sources.
- Clock output function with divider that can reflect the system oscillator clock, IRC clock, CPU clock, or the watchdog clock.
- Processor wake-up from Deep-sleep mode via a dedicated start logic using up to 40 of the functional pins.
- Brownout detect with four separate thresholds for interrupt and one threshold for forced reset (four thresholds for forced reset on the LPC1311/01 and LPC1313/01 parts).
- Power-On Reset (POR).
- Integrated oscillator with an operating range of 1 MHz to 25 MHz.
- 12 MHz internal RC oscillator trimmed to 1 % accuracy over the entire temperature and voltage range that can optionally be used as a system clock.
- Programmable watchdog oscillator with a frequency range of 7.8 kHz to 1.8 MHz.

- System 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.
- For USB (LPC1342/43), a second, dedicated PLL is provided.
- Code Read Protection (CRP) with different security levels.
- Unique device serial number for identification.
- Available as 48-pin LQFP package and 33-pin HVQFN package.

1.4 Ordering options

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
LPC1311FHN33	HVQFN33	HVQFN33: plastic thermal enhanced very thin quad flat package; no leads; 33 terminals; body 7 × 7 × 0.85 mm	n/a
LPC1311FHN33/01	HVQFN33	HVQFN33: plastic thermal enhanced very thin quad flat package; no leads; 33 terminals; body 7 × 7 × 0.85 mm	n/a
LPC1313FHN33	HVQFN33	HVQFN33: plastic thermal enhanced very thin quad flat package; no leads; 33 terminals; body 7 × 7 × 0.85 mm	n/a
LPC1313FHN33/01	HVQFN33	HVQFN33: plastic thermal enhanced very thin quad flat package; no leads; 33 terminals; body 7 × 7 × 0.85 mm	n/a
LPC1313FBD48	LQFP48	LQFP48: plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2
LPC1313FBD48/01	LQFP48	LQFP48: plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2
LPC1342FHN33	HVQFN33	HVQFN33: plastic thermal enhanced very thin quad flat package; no leads; 33 terminals; body 7 × 7 × 0.85 mm	n/a
LPC1342FBD48	LQFP48	LQFP48: plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2
LPC1343FHN33	HVQFN33	HVQFN33: plastic thermal enhanced very thin quad flat package; no leads; 33 terminals; body 7 × 7 × 0.85 mm	n/a
LPC1343FBD48	LQFP48	LQFP48: plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2

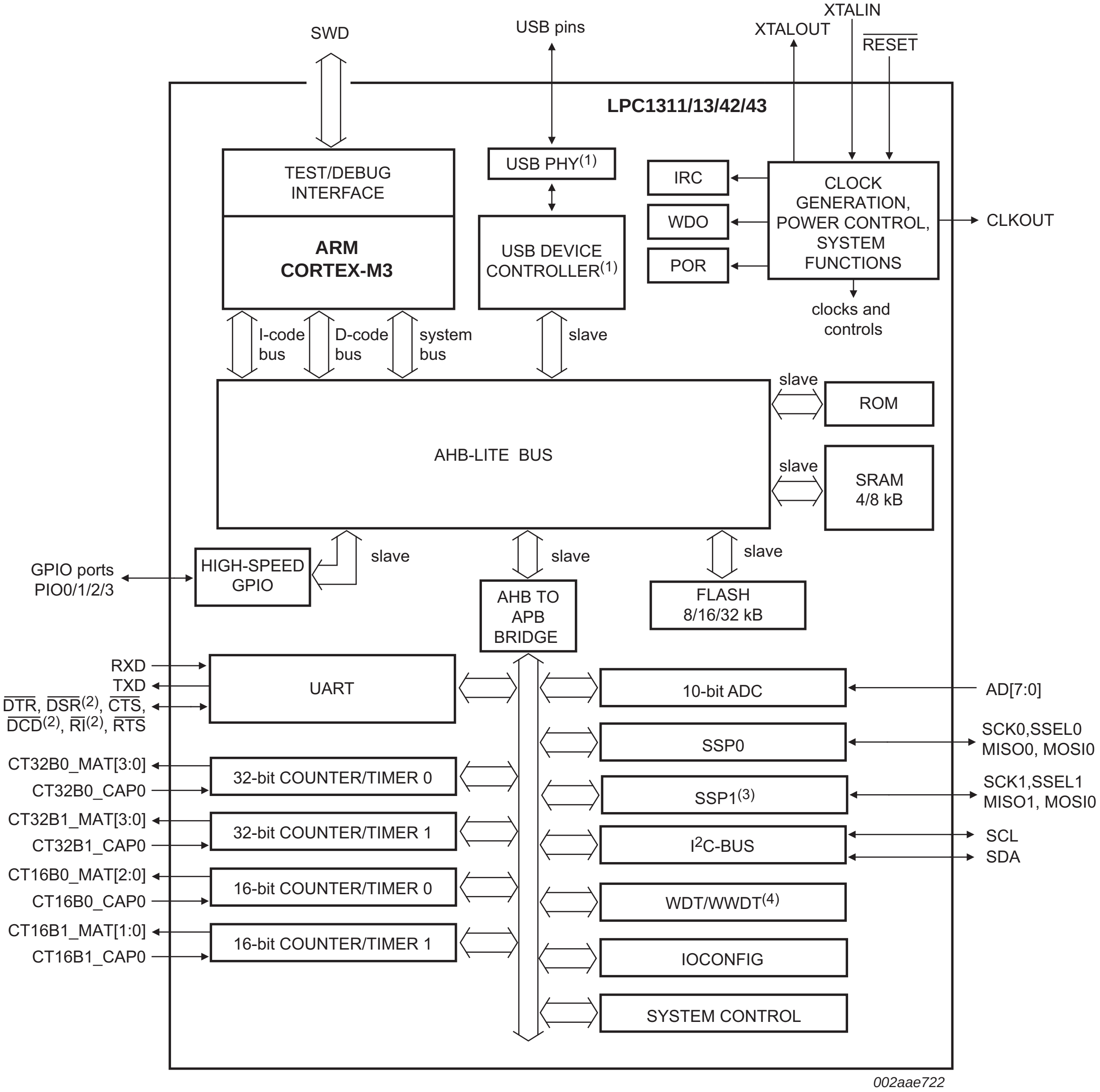
Table 2. Ordering options for LPC13xx

Type number	Flash	Total SRAM	USB	Power profiles	UART RS-485	I^2C/ Fast+	SSP	ADC channels	Pins	Package
LPC1311FHN33	8 kB	4 kB	-	no	1	1	1	8	33	HVQFN33
LPC1311FHN33/01	8 kB	4 kB	-	yes	1	1	1	8	33	HVQFN33
LPC1313FHN33	32 kB	8 kB	-	no	1	1	1	8	33	HVQFN33
LPC1313FHN33/01	32 kB	8 kB	-	yes	1	1	1	8	33	HVQFN33
LPC1313FBD48	32 kB	8 kB	-	no	1	1	1	8	48	LQFP48
LPC1313FBD48/01	32 kB	8 kB	-	yes	1	1	2	8	48	LQFP48
LPC1342FHN33	16 kB	4 kB	Device	no	1	1	1	8	33	HVQFN33

Table 2. Ordering options for LPC13xx

Type number	Flash	Total SRAM	USB	Power profiles	UART RS-485	I^2C/ Fast+	SSP	ADC channels	Pins	Package
LPC1342FBD48	16 kB	4 kB	Device	no	1	1	1	8	48	LQFP48
LPC1343FHN33	32 kB	8 kB	Device	no	1	1	1	8	33	HVQFN33
LPC1343FBD48	32 kB	8 kB	Device	no	1	1	1	8	48	LQFP48

1.5 Block diagram



(1) LPC1342/43 only.

(2) LQFP48 package only.

(3) On LPC1313FBD48/01 only.

(4) Windowed WatchDog Timer (WWDT) on LPC1311/01 and LPC1313/01 only.

Fig 1. LPC13xx block diagram

2.1 How to read this chapter

See Table 3 for LPC13xx memory configurations:

Table 3. LPC13xx memory configuration

Part	Flash	Address range	SRAM	Address range
LPC1311	8 kB	0x0000 0000 - 0x0000 1FFF	4 kB	0x1000 0000 - 0x1000 0FFF
LPC1311/01	8 kB	0x0000 0000 - 0x0000 1FFF	4 kB	0x1000 0000 - 0x1000 0FFF
LPC1313	32 kB	0x0000 0000 - 0x0000 7FFF	8 kB	0x1000 0000 - 0x1000 1FFF
LPC1313/01	32 kB	0x0000 0000 - 0x0000 7FFF	8 kB	0x1000 0000 - 0x1000 1FFF
LPC1342	16 kB	0x0000 0000 - 0x0000 3FFF	4 kB	0x1000 0000 - 0x1000 0FFF
LPC1343	32 kB	0x0000 0000 - 0x0000 7FFF	8 kB	0x1000 0000 - 0x1000 1FFF

2.2 Memory map

Figure 2 shows the memory and peripheral address space of the LPC13xx.

The AHB peripheral area is 2 MB in size and is divided to allow for up to 128 peripherals. On the LPC13xx, the GPIO ports are the only 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.

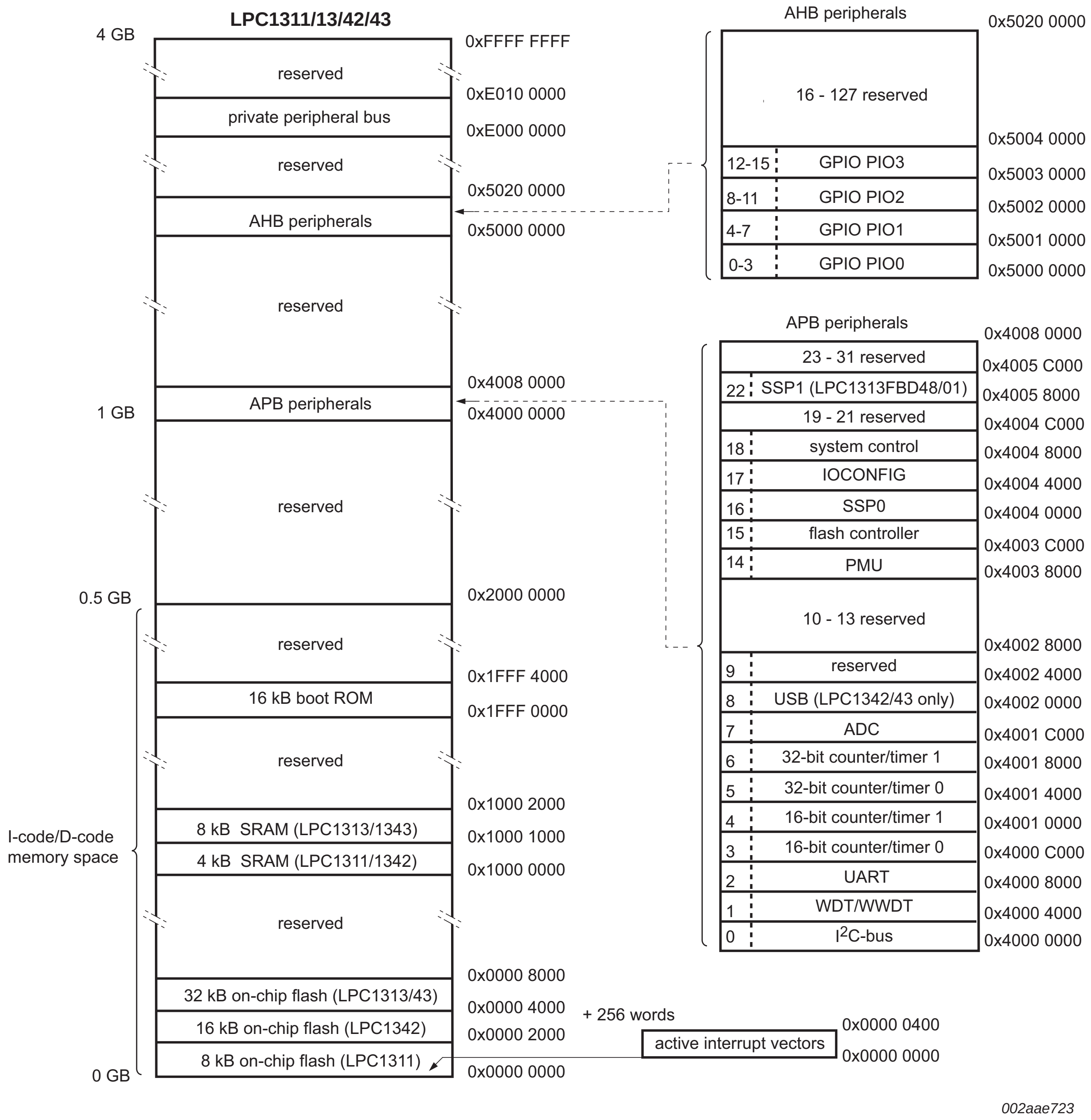


Fig 2. LPC13xx memory map

2.3 Memory remapping

For details, see Table 8.

UM10375

Chapter 3: LPC13xx System configuration

Rev. 5 — 21 June 2012 User manual

3.1 How to read this chapter

The system configuration registers apply to all LPC13xx parts with the following exceptions:

USB clocking and power control

Since the USB block is available on the LPC1342 and LPC1343 only, the registers and register bits listed in Table 4 are reserved for parts LPC1311 and LPC1313:

Table 4. USB related registers and register bits reserved for LPC1311/13

Name	Access	Address offset	Description	Register bits reserved for LPC1311/13
USBPLLCTRL	R/W	0x010	USB PLL control	all
USBPLLSTAT	R	0x014	USB PLL status	all
USBPLLCLKSEL	R/W	0x048	USB PLL clock source select	all
USBPLLCLKUEN	R/W	0x04C	USB PLL clock source update enable	all
SYSAHBCLKCTRL	R/W	0x080	System AHB clock control	bit 14
USBCLKSEL	R/W	0x0C0	USB clock source select	all
USBCLKUEN	R/W	0x0C4	USB clock source update enable	all
USBCLKDIV	R/W	0x0C8	USB clock source divider	all
PDSLEEPCFG	R/W	0x230	Power-down states in Deep-sleep mode	bits 8 and 10
PDAWAKECFG	R/W	0x234	Power-down states after wake-up from Deep-sleep mode	bits 8 and 10
PDRUNCFG	R/W	0x238	Power-down configuration register	bits 8 and 10

SSP1

The SSP1 block is available on the LPC1313FBD48/01 only. SSP1 related registers and register bits are reserved for the following parts: LPC1311/13/42/43 and LPC1311FHN33/01 and LPC1313FHN33/01.

BOD control

The number of programmable BOD levels for forced reset is different for the LPC1300 and the LPC1300L series. See Table 5. The BOD trip levels for the LPC1300 and LPC1300L series are listed in the *LPC1311/13/42/43 data sheet*.

Table 5. BOD interrupt and reset levels

Series	Type number	Interrupt levels	Reset levels
LPC1300	LPC1311FHN33	4 (programmable)	1 (fixed)
LPC1300	LPC1313FBD48	4 (programmable)	1 (fixed)
LPC1300	LPC1313FHN33	4 (programmable)	1 (fixed)
LPC1300	LPC1342FHN33	4 (programmable)	1 (fixed)
LPC1300	LPC1343FBD48	4 (programmable)	1 (fixed)

Table 5. BOD interrupt and reset levels

Series	Type number	Interrupt levels	Reset levels
LPC1300	LPC1343FHN33	4 (programmable)	1 (fixed)
LPC1300L	LPC1311FHN33/01	4 (programmable)	4 (programmable)
LPC1300L	LPC1313FHN33/01	4 (programmable)	4 (programmable)
LPC1300L	LPC1313FBD48/01	4 (programmable)	4 (programmable)

Input pins to the start logic

For HVQFN packages, the start logic control bits (see Table 44 to Table 51) are reserved for port pins PIO2_1 to PIO2_11 and PIO3_0, PIO3_1, and PIO3_3.

PIO reset status registers

For HVQFN packages, the reset status bits (see Table 40 and Table 41) are reserved for port pins PIO2_1 to PIO2_11 and PIO3_0 and PIO3_1, and PIO3_3.

Entering Deep power-down mode

Status of the IRC before entering Deep power-down mode (see Section 3.9.4.2):

- IRC must be enabled for parts LPC1311/13/42/43.
- IRC status has no effect for parts LPC1311/01 and LPC1313/01.

Enabling sequence for UART clock

Requirements for enabling the UART peripheral clock:

- The UART pins must be configured in the IOCON block before the UART clock can be enabled in the in the SYSAHBCLKCTRL register (Table 25) for parts LPC1311/13/42/43.
- The sequence of configuring the UART pins and the UART clock has no effect for parts LPC1311/01 and LPC1313/01.

Deep-sleep mode configuration

Register values configuring the Deep-sleep mode are different for the LPC1300 (parts LPC1311/13/42/43) and LPC1300L (parts LPC1311/01 and LPC1313/01) series (see Section 3.5.45, the PDSLEEPCFG register).

3.2 Introduction

The system configuration block controls oscillators, the power management unit, and clock generation of the LPC13xx. 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.

3.3 Pin description

Table 6 shows pins that are associated with system control block functions.

Table 6. Pin summary

Pin name	Pin direction	Pin description
CLKOUT	O	Clockout pin
PIO0_0 to PIO0_11	I	Wake-up pins port 0
PIO1_0 to PIO1_11	I	Wake-up pins port 1
PIO2_0 to PIO2_11[1]	I	Wake-up pins port 2
PIO3_0 to PIO3_3[1]	I	Wake-up pins port 3

[1] For HVQFN packages, applies to P2_0, P3_2, and P3_3 only.

3.4 Clocking and power control

See Figure 3 for an overview of the LPC13xx Clock Generation Unit (CGU).

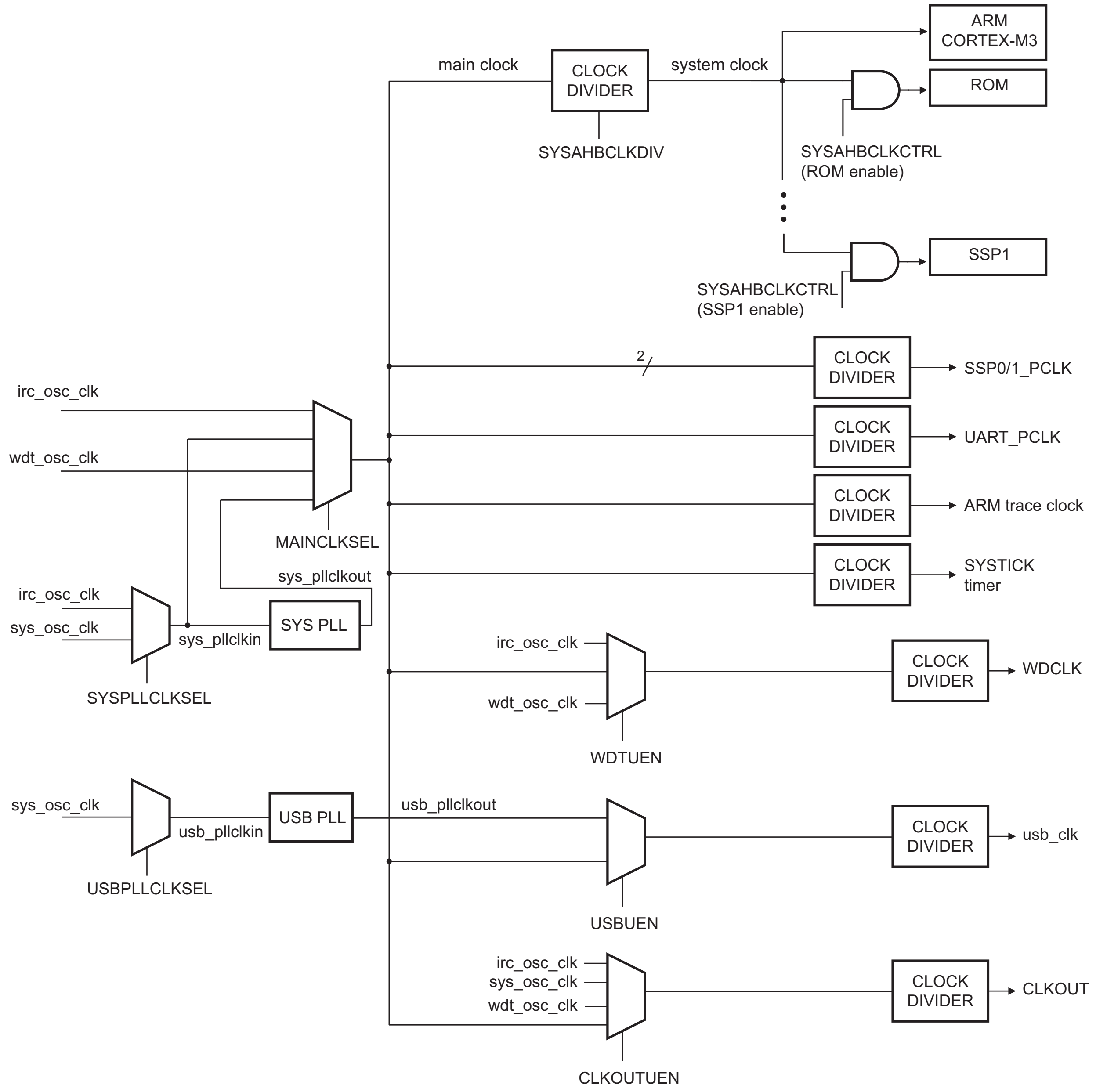
The LPC131x include three independent oscillators. These are the system oscillator, the Internal RC oscillator (IRC), and the Watchdog oscillator. Each oscillator can be used for more than one purpose as required in a particular application.

Following reset, the LPC131x will operate from the Internal RC oscillator until switched by software. This allows systems to operate without any external crystal and the bootloader code to operate at a known frequency.

The SYSAHBCLKCTRL register gates the system clock to the various peripherals and memories. UART, SSP0/1, the SysTick timer, and the ARM trace clock have individual clock dividers to derive peripheral clocks from the main clock.

The USB clock, if available, and 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.



USB is available in parts LPC134x only.

SSP1 is available on part LPC1313FBD48 only.

Fig 3. LPC13xx CGU block diagram

3.5 Register description

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

See Section 3.12 for the flash access timing register, which can be re-configured as part the system setup. This register is not part of the system configuration block.

Table 7. Register overview: system control block (base address 0x4004 8000)

Name	Access	Address offset	Description	Reset value	Reference
SYSMEMREMAP	R/W	0x000	System memory remap	0x0000 0002	Table 8
PRESETCTRL	R/W	0x004	Peripheral reset control	0x0000 0000	Table 9
SYSPLLCTRL	R/W	0x008	System PLL control	0x0000 0000	Table 10
SYSPLLSTAT	R	0x00C	System PLL status	0x0000 0000	Table 11
USBPLLCTRL	R/W	0x010	USB PLL control	0x0000 0000	Table 12
USBPLLSTAT	R	0x014	USB PLL status	0x0000 0000	Table 13
-	-	0x018 - 0x01C	Reserved	-	-
SYSOSCCTRL	R/W	0x020	System oscillator control	0x0000 0000	Table 14
WDTOSCCTRL	R/W	0x024	Watchdog oscillator control	0x0000 0000	Table 15
IRCCTRL	R/W	0x028	IRC control	0x0000 0080	Table 16
-	-	0x02C	Reserved	-	-
SYSRESSTAT	R/W	0x030	System reset status register	0x0000 0000	Table 17
-	-	0x034 - 0x03C	Reserved	-	-
SYSPLLCLKSEL	R/W	0x040	System PLL clock source select	0x0000 0000	Table 18
SYSPLLCLKUEN	R/W	0x044	System PLL clock source update enable	0x0000 0000	Table 19
USBPLLCLKSEL	R/W	0x048	USB PLL clock source select	0x0000 0000	Table 20
USBPLLCLKUEN	R/W	0x04C	USB PLL clock source update enable	0x0000 0000	Table 21
-	-	0x050 - 0x06C	Reserved	-	-
MAINCLKSEL	R/W	0x070	Main clock source select	0x0000 0000	Table 22
MAINCLKUEN	R/W	0x074	Main clock source update enable	0x0000 0000	Table 23
SYSAHBCLKDIV	R/W	0x078	System AHB clock divider	0x0000 0001	Table 24
-	-	0x07C	Reserved	-	-
SYSAHBCLKCTRL	R/W	0x080	System AHB clock control	0x0000 485F	Table 25
-	-	0x084 - 0x090	Reserved	-	-
SSP0CLKDIV	R/W	0x094	SSP0 clock divider	0x0000 0000	Table 26
UARTCLKDIV	R/W	0x098	UART clock divder	0x0000 0000	Table 27
SSP1CLKDIV	R/W	0x09C	SSP1 clock divider	0x000	Table 28
-	-	0x0A0 - 0x0A8	Reserved	-	-
TRACECLKDIV	R/W	0x0AC	ARM trace clock divider	0x0000 0000	Table 29
SYSTICKCLKDIV	R/W	0x0B0	SYSTICK clock divder	0x0000 0000	Table 30
-	-	0x0B4 - 0x0BC	Reserved	-	-
USBCLKSEL	R/W	0x0C0	USB clock source select	0x0000 0000	Table 31
USBCLKUEN	R/W	0x0C4	USB clock source update enable	0x0000 0000	Table 32
USBCLKDIV	R/W	0x0C8	USB clock source divider	0x0000 0000	Table 33
-	-	0x0CC	Reserved	-	-
WDTCLKSEL	R/W	0x0D0	WDT clock source select	0x0000 0000	Table 34
WDTCLKUEN	R/W	0x0D4	WDT clock source update enable	0x0000 0000	Table 35
WDTCLKDIV	R/W	0x0D8	WDT clock divider	0x0000 0000	Table 36
-	-	0x0DC	Reserved	-	-
CLKOUTCLKSEL	R/W	0x0E0	CLKOUT clock source select	0x0000 0000	Table 37
CLKOUTUEN	R/W	0x0E4	CLKOUT clock source update enable	0x0000 0000	Table 38

Table 7. Register overview: system control block (base address 0x4004 8000) *…continued*

Name	Access	Address offset	Description	Reset value	Reference
CLKOUTDIV	R/W	0x0E8	CLKOUT clock divider	0x0000 0000	Table 39
-	-	0x0EC - 0x0FC	Reserved	-	-
PIOPORCAP0	R	0x100	POR captured PIO status 0	-	Table 40
PIOPORCAP1	R	0x104	POR captured PIO status 1	-	Table 40
-	-	0x108 - 0x14C	Reserved	0x0000 0000	-
BODCTRL	R/W	0x150	BOD control	0x0000 0000	Table 42
SYSTCKCAL	R/W	0x154	System tick counter calibration	0x0000 0004	Table 43
-	-	0x158 - 0x1FC	Reserved	-	-
STARTAPRP0	R/W	0x200	Start logic edge control register 0; bottom 32 interrupts	-	Table 44
STARTERP0	R/W	0x204	Start logic signal enable register 0; bottom 32 interrupts	-	Table 45
STARTRSRP0CLR	W	0x208	Start logic reset register 0; bottom 32 interrupts	-	Table 46
STARTSRP0	R	0x20C	Start logic status register 0; bottom 32 interrupts	-	Table 47
STARTAPRP1	R/W	0x210	Start logic edge control register 1; top 8 interrupts	-	Table 48
STARTERP1	R/W	0x214	Start logic signal enable register 1; top 8 interrupts	-	Table 49
STARTRSRP1CLR	W	0x218	Start logic reset register 1; top 8 interrupts	-	Table 50
STARTSRP1	R	0x21C	Start logic status register 1; top 8 interrupts	-	Table 51
-	-	0x220 - 0x22C	Reserved	-	-
PDSLEEPCFG	R/W	0x230	Power-down states in Deep-sleep mode	0x0000 0000	Table 53
PDAWAKECFG	R/W	0x234	Power-down states after wake-up from Deep-sleep mode	0x0000 FDF0	Table 54
PDRUNCFG	R/W	0x238	Power-down configuration register	0x0000 FDF0	Table 55
-	-	0x23C - 0x3F0	Reserved	-	-
DEVICE_ID	R	0x3F4	Device ID	part dependent	Table 56

3.5.1 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	10
		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

3.5.2 Peripheral reset control register

This register allows software to reset the SSP0/1 and I2C peripherals. Writing a 0 to the SSP0/1_RST_N or I2C_RST_N bits resets the SSP0/1 or I2C peripherals. Writing a 1 de-asserts the reset.

Remark: Before accessing the SSP0/1 and I2C peripherals, write a 1 to this register to ensure that the reset signals to the SSP0/1 and I2C are de-asserted.

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

Bit	Symbol	Value	Description	Reset value
0	SSP0_RST_N		SSP0 reset control	0
		0	Reset SSP0.	
		1	De-assert SSP0 reset.	
1	I2C_RST_N		I2C reset control	0
		0	Reset I2C.	
		1	De-asset I2C reset.	
2	SSP1_RST_N		SSP1 reset control	0
		0	Reset the SSP1.	
		1	De-assert SSP1 reset.	
31:3	-	-	Reserved	0x00

3.5.3 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 optionally the USB subsystem. Note that the USB subsystem has its own dedicated PLL. The PLL can produce a clock up to the maximum allowed for the CPU, which is 72 MHz.

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.	0x000
6:5	PSEL		Post divider ratio P. The division ratio is 2 × P.	0x00
		0x0	P = 1	
		0x1	P = 2	
		0x2	P = 4	
		0x3	P = 8	
31:7	-	-	Reserved. Do not write ones to reserved bits.	0x00

3.5.4 System PLL status register

This register is a Read-only register and supplies the PLL lock status (see Section 3.11.1).

Table 11. System PLL status register (SYSPLLSTAT, address 0x4004 800C) bit description

Bit	Symbol	Value	Description	Reset value
0	LOCK		PLL lock status	0x0
		0	PLL not locked	
		1	PLL locked	
31:1	-	-	Reserved	0x00

3.5.5 USB PLL control register

The USB PLL is identical to the system PLL and is used to provide a dedicated clock to the USB block if available (see Section 3.1).

This register connects and enables the USB 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 48 MHz clock used by the USB subsystem.

Remark: The USB PLL must be connected to the system oscillator for correct USB operation (see Table 20).

Table 12. USB PLL control register (USBPLLCTRL, address 0x4004 8010) 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.	0x000

Table 12. USB PLL control register (USBPLLCTRL, address 0x4004 8010) bit description

Bit	Symbol	Value	Description	Reset value
6:5	PSEL		Post divider ratio P. The division ratio is 2 × P.	0x00
		0x0	P = 1	
		0x1	P = 2	
		0x2	P = 4	
		0x3	P = 8	
31:7	-	-	Reserved. Do not write ones to reserved bits.	0x00

3.5.6 USB PLL status register

This register is a Read-only register and supplies the PLL lock status (see Section 3.11.1).

Table 13. USB PLL status register (USBPLLSTAT, address 0x4004 8014) bit description

Bit	Symbol	Value	Description	Reset value
0	LOCK		PLL lock status	0x0
		0	PLL not locked	
		1	PLL locked	
31:1	-	-	Reserved	0x00

3.5.7 System oscillator control register

This register configures the frequency range for the system oscillator.

Table 14. System oscillator control register (SYSOSCCTRL, address 0x4004 8020) bit description

Bit	Symbol	Value	Description	Reset value
0	BYPASS		Bypass system oscillator	0x0
		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.	0x0
		0	1 - 20 MHz frequency range.	
		1	15 - 25 MHz frequency range	
31:2	-	-	Reserved	0x00

3.5.8 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 × (1 + DIVSEL)) = 7.8 kHz to 1.7 MHz (nominal values).

Remark: Any setting of the FREQSEL bits will yield a Fclkana value within ± 40% of the listed frequency value. The watchdog oscillator is the clock source with the lowest power consumption. If accurate timing is required, use the IRC or system 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 15. 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	0x0

Table 15. Watchdog oscillator control register (WDTOSCCTRL, address 0x4004 8024) bit description

Bit	Symbol	Value	Description	Reset value
8:5	FREQSEL		Select watchdog oscillator analog output frequency (Fclkana).	0x00
		0x1	0.6 MHz	
		0x2	1.05 MHz	
		0x3	1.4 MHz	
		0x4	1.75 MHz	
		0x5	2.1 MHz	
		0x6	2.4 MHz	
		0x7	2.7 MHz	
		0x8	3.0 MHz	
		0x9	3.25 MHz	
		0xA	3.5 MHz	
		0xB	3.75 MHz	
		0xC	4.0 MHz	
		0xD	4.2 MHz	
		0xE	4.4 MHz	
		0xF	4.6 MHz	
31:9	-	-	Reserved	0x00

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

Bit	Symbol	Description	Reset value
7:0	TRIM	Trim value	0x1000 0000, then flash will reprogram
31:8	-	Reserved	0x00

3.5.10 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 17. System reset status register (SYSRESSTAT, address 0x4004 8030) bit description

Bit	Symbol	Value	Description	Reset value
0	POR		POR reset status	0x0
		0	No POR detected	
		1	POR detected	
1	EXTRST			0x0
		0	No $\overline{\text{RESET}}$ event detected	
		1	$\overline{\text{RESET}}$ detected	
2	WDT		Status of the Watchdog reset	0x0
		0	No WDT reset detected	
		1	WDT reset detected	
3	BOD		Status of the Brown-out detect reset	0x0
		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.	0x0
		0	No System reset detected	
		1	System reset detected	
31:5	-	-	Reserved	0x00

3.5.11 System PLL clock source select register

This register selects the clock source for the system PLL. The SYSPLLCLKUEN register (see Section 3.5.12) must be toggled from LOW to HIGH for the update to take effect.

Remark: The system oscillator must be selected if the system PLL is used to generate a 48 MHz clock to the USB block.

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

Table 18. 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

3.5.12 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.

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

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

3.5.13 USB PLL clock source select register

This register selects the clock source for the dedicated USB PLL. The USBPLLCLKUEN register (see Section 3.5.14) must be toggled from LOW to HIGH for the update to take effect.

Remark: When switching clock sources, both clocks must be running before the clock source is updated in the USBPLLCLKUEN register. For USB operation, the clock source must be switched from IRC to system oscillator with both the IRC and the system oscillator running. After the switch, the IRC can be turned off.

Table 20. USB PLL clock source select register (USBPLLCLKSEL, address 0x4004 8048) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		USB PLL clock source	0x00
		0x0	IRC. The USB PLL clock source must be switched to system oscillator for correct USB operation.	
		0x1	System oscillator	
		0x2	Reserved	
		0x3	Reserved	
31:2	-	-	Reserved	0x00

3.5.14 USB PLL clock source update enable register

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

Remark: The system oscillator must be selected in the USBPLLCLKSEL register in order to use the USB PLL, and this register must be toggled to update the USB PLL clock with the system oscillator.

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

Table 21. USB PLL clock source update enable register (USBPLLCLKUEN, address 0x4004 804C) bit description

Bit	Symbol	Value	Description	Reset value
0	ENA		Enable USB PLL clock source update	0x0
		0	No change	
		1	Update clock source	
31:1	-	-	Reserved	0x00

3.5.15 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 memories, and optionally the USB block.

The MAINCLKUEN register (see Section 3.5.16) must be toggled from LOW to HIGH for the update to take effect.

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

Table 22. 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	0x00
		0x0	IRC oscillator	
		0x1	Input clock to system PLL	
		0x2	WDT oscillator	
		0x3	System PLL clock out	
31:2	-	-	Reserved	0x00

3.5.16 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.

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

Table 23. 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

3.5.17 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 24. 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.	0x01
31:8	-	Reserved	0x00

3.5.18 System AHB clock control register

The SYSAHBCLKCTRL register enables the clocks to individual system and peripheral blocks. The system clock (sys_ahb_clk[0], bit 0 in the SYSAHBCLKCTRL register) provides the clock for the AHB to APB bridge, the AHB matrix, the ARM Cortex-M3, the Syscon block, and the PMU. This clock cannot be disabled.

Table 25. 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-M3 FCLK and HCLK, to the SysCon, and to the PMU. This bit is read only.	1
		0	Reserved	
		1	Enabled	
1	ROM		Enables clock for ROM.	1
		0	Disabled	
		1	Enabled	
2	RAM		Enables clock for RAM.	1
		0	Disabled	
		1	Enabled	

Table 25. System AHB clock control register (SYSAHBCLKCTRL, address 0x4004 8080) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
3	FLASHREG		Enables clock for flash register interface.	1
		0	Disabled	
		1	Enabled	
4	FLASHARRAY		Enables clock for flash array access.	1
		0	Disabled	
		1	Enabled	
5	I2C		Enables clock for I2C.	0
		0	Disabled	
		1	Enabled	
6	GPIO		Enables clock for GPIO.	1
		0	Disabled	
		1	Enabled	
7	CT16B0		Enables clock for 16-bit counter/timer 0.	0
		0	Disabled	
		1	Enabled	
8	CT16B1		Enables clock for 16-bit counter/timer 1.	0
		0	Disabled	
		1	Enabled	
9	CT32B0		Enables clock for 32-bit counter/timer 0.	0
		0	Disabled	
		1	Enabled	
10	CT32B1		Enables clock for 32-bit counter/timer 1.	0
		0	Disabled	
		1	Enabled	
11	SSP		Enables clock for SSP.	1
		0	Disabled	
		1	Enabled	
12	UART		Enables clock for UART. Note that for the LPC1311/13/42/43, the UART pins must be configured in the IOCON block before the UART clock can be enabled. For the LPC1311/01 and LPC1313/01 no special enabling sequence is required.	0
		0	Disabled	
		1	Enabled	
13	ADC		Enables clock for ADC.	0
		0	Disabled	
		1	Enabled	
14	USB_REG		Enables clock for USB_REG.	1
		0	Disabled	
		1	Enabled	

Table 25. System AHB clock control register (SYSAHBCLKCTRL, address 0x4004 8080) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
15	WDT		Enables clock for WDT.	0
		0	Disabled	
		1	Enabled	
16	IOCON		Enables clock for IO configuration block.	0
		0	Disabled	
		1	Enabled	
17	-	-	Reserved	0x00
18	SSP1		Enables clock for SSP1.	0
		0	Disable	
		1	Enable	
31:19	-	-	Reserved	0x00

3.5.19 SSP0 clock divider register

This register configures the SSP0 peripheral clock SSP0_PCLK. The SSP0_PCLK can be shut down by setting the DIV bits to 0x0.

Table 26. SSP0 clock divider register (SSP0CLKDIV, address 0x4004 8094) bit description

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

3.5.20 UART clock divider register

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

Remark: Note that for the LPC1311/13/42/43, the UART pins must be configured in the IOCON block before the UART clock can be enabled. For the LPC1311/01 and LPC1313/01 no special enabling sequence is required.

Table 27. UART clock divider register (UARTCLKDIV, address 0x4004 8098) bit description

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

3.5.21 SSP1 clock divider register

This register configures the SSP1 peripheral clock SSP1_PCLK. The SSP1_PCLK can be shut down by setting the DIV bits to 0x0.

Table 28. SSP1 clock divider register (SSP1CLKDIV, address 0x4004 809C) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	SSP1_PCLK clock divider values 0: Disable SSP1_PCLK. 1: Divide by 1. to 255: Divide by 255.	0x00
31:8	-	Reserved	0x00

3.5.22 Trace clock divider register

This register configures the ARM trace clock. The trace clock can be shut down by setting the DIV bits to 0x0.

Table 29. TRACECLKDIV clock divider register (TRACECLKDIV, address 0x4004 80AC) bit description

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

3.5.23 SYSTICK clock divider register

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

Table 30. SYSTICK clock divider register (SYSTICKCLKDIV, address 0x4004 80B0) bit description

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

3.5.24 USB clock source select register

This register selects the clock source for the USB usb_clk. The clock source can be either the USB PLL output or the main clock, and the clock can be further divided by the USBCLKDIV register (see Table 33) to obtain a 48 MHz clock.

The USBCLKUEN register (see Section 3.5.25) must be toggled from LOW to HIGH for the update to take effect.

Remark: When switching clock sources, both clocks must be running before the clock source is updated. The default clock source for the USB controller is the USB PLL output. For switching the clock source to the main clock, ensure that the system PLL and the USB PLL are running to make both clock sources available for switching. The main clock must be set to 48 MHz and configured with the main PLL and the system oscillator. After the switch, the USB PLL can be turned off.

Table 31. USB clock source select register (USBCLKSEL, address 0x4004 80C0) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		USB clock source	0x00
		0x0	USB PLL out	
		0x1	Main clock	
		0x2	Reserved	
		0x3	Reserved	
31:2	-	-	Reserved	0x00

3.5.25 USB clock source update enable register

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

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

Table 32. USB clock source update enable register (USBCLKUEN, address 0x4004 80C4) bit description

Bit	Symbol	Value	Description	Reset value
0	ENA		Enable USB clock source update	0x0
		0	No change	
		1	Update clock source	
31:1	-	-	Reserved	0x00

3.5.26 USB clock divider register

This register allows the USB clock usb_clk to be divided to 48 MHz. The usb_clk can be shut down by setting the DIV bits to 0x0.

Table 33. USB clock divider register (USBCLKDIV, address 0x4004 80C8) bit description

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

3.5.27 WDT clock source select register

This register selects the clock source for the watchdog timer. The WDTCLKUEN register (see Section 3.5.28) must be toggled from LOW to HIGH for the update to take effect.

Remark: When switching clock sources, both clocks must be running before the clock source is updated. Once the WWDT (LPC1311/01 and LPC1313/01 only) is enabled, the watchdog clock source cannot be changed. If the watchdog timer is running in Deep-sleep mode, always select the watchdog oscillator as clock source (see Section 3.9.3.2).

Table 34. WDT clock source select register (WDTCLKSEL, address 0x4004 80D0) bit description

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

3.5.28 WDT clock source update enable register

This register updates the clock source of the watchdog timer with the new input clock after the WDTCLKSEL register has been written to. In order for the update to take effect at the input of the watchdog timer, first write a zero to the WDTCLKUEN register and then write a one to WDTCLKUEN.

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

Table 35. WDT clock source update enable register (WDTCLKUEN, address 0x4004 80D4) bit description

Bit	Symbol	Value	Description	Reset value
0	ENA		Enable WDT clock source update	0x0
		0	No change	
		1	Update clock source	
31:1	-	-	Reserved	0x00

3.5.29 WDT clock divider register

This register determines the divider values for the watchdog clock wdt_clk.

Table 36. WDT clock divider register (WDTCLKDIV, address 0x4004 80D8) bit description

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

3.5.30 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 3.5.31) must be toggled from LOW to HIGH for the update to take effect.

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

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

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

3.5.31 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 CLKCLKUEN register and then write a one to CLKCLKUEN.

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

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

Bit	Symbol	Value	Description	Reset value
0	ENA		Enable CLKOUT clock source update	0x0
		0	No change	
		1	Update clock source	
31:1	-	-	Reserved	0x00

3.5.32 CLKOUT clock divider register

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

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

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

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

Bit	Symbol	Description	Reset value
11:0	CAPPIO0_11 to CAPPIO0_0	Raw reset status input PIO0_11 to PIO0_0	User implementation dependent
23:12	CAPPIO1_11 to CAPPIO1_0	Raw reset status input PIO1_11 to PIO1_0	User implementation dependent
31:24	CAPPIO2_7 to CAPPIO2_0	Raw reset status input PIO2_7 to PIO2_0	User implementation dependent

3.5.34 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) and port 3 at power-on-reset. Each bit represents the reset state of one PIO pin. This register is a read-only status register.

Table 41. POR captured PIO status registers 1 (PIOPORCAP1, address 0x4004 8104) bit description

Bit	Symbol	Description	Reset value
0	CAPPIO2_8	Raw reset status input PIO2_8	User implementation dependent
1	CAPPIO2_9	Raw reset status input PIO2_9	User implementation dependent
2	CAPPIO2_10	Raw reset status input PIO2_10	User implementation dependent
3	CAPPIO2_11	Raw reset status input PIO2_11	User implementation dependent
4	CAPPIO3_0	Raw reset status input PIO3_0	User implementation dependent
5	CAPPIO3_1	Raw reset status input PIO3_1	User implementation dependent
6	CAPPIO3_2	Raw reset status input PIO3_2	User implementation dependent
7	CAPPIO3_3	Raw reset status input PIO3_3	User implementation dependent
8	CAPPIO3_4	Raw reset status input PIO3_4	User implementation dependent
9	CAPPIO3_5	Raw reset status input PIO3_5	User implementation dependent
31:10	-	Reserved	-

3.5.35 BOD control register

The BOD control register selects four separate threshold values for sending a BOD interrupt to the NVIC. Only one level is allowed for forced reset.

Table 42. BOD control register (BODCTRL, address 0x4004 8150) bit description

Bit	Symbol	Value	Description	Reset value
1:0	BODRSTLEV		BOD reset level. Trip values x/y refer to the LPC1300/LPC1300L series.	0
		0x0	The reset assertion threshold voltage is 1.49 V/1.46 V; the reset de-assertion threshold voltage is 1.64 V/1.63 V.	
		0x1	The reset assertion threshold voltage is -/2.06 V; the reset de-assertion threshold voltage is -/2.15 V.	
		0x2	The reset assertion threshold voltage is -/2.35 V; the reset de-assertion threshold voltage is -/2.43 V.	
		0x3	The reset assertion threshold voltage is -/2.63 V; the reset de-assertion threshold voltage is -/2.71 V.	
3:2	BODINTVAL		BOD interrupt level. Trip values x/y refer to the LPC1300/LPC1300L series.	0
		0x0	The interrupt assertion threshold voltage is 1.69 V/1.65 V; the interrupt de-assertion threshold voltage is 1.84 V/1.8 V.	
		0x1	The interrupt assertion threshold voltage is 2.29 V/2.22 V; the interrupt de-assertion threshold voltage is 2.44 V/2.35 V.	
		0x2	The interrupt assertion threshold voltage is 2.59 V/ 2.52 V; the interrupt de-assertion threshold voltage is 2.74 V/2.66 V.	
		0x3	The interrupt assertion threshold voltage is 2.87 V/2.80 V; the interrupt de-assertion threshold voltage is 2.98 V/2.90 V.	
4	BODRSTENA		BOD reset enable	0
		0	Disable reset function.	
		1	Enable reset function.	
31:5	-	-	Reserved	0

3.5.36 System tick counter calibration register

This register determines the value of the SYST_CALIB register (see Table 288).

Table 43. System tick timer calibration register (SYSTCKCAL, address 0x4004 8154) bit description

Bit	Symbol	Description	Reset value
25:0	CAL	System tick timer calibration value	0x04
31:26	-	Reserved	0x00

3.5.37 Start logic edge control register 0

The STARTAPRP0 register controls the start logic inputs of ports 0 (PIO0_0 to PIO0_11) and 1 (PIO1_0 to PIO1_11) and the lower 8 inputs of port 2 (PIO2_0 to PIO2_7). 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 3.10.2).

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, etc. (see Table 66). The bottom 32 interrupts are contained this register, the top 8 interrupts are contained in the STARTAPRP1 register for total of 40 wake-up interrupts.

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 chip from Deep-sleep mode.

Table 44. Start logic edge control register 0 (STARTAPRP0, address 0x4004 8200) bit description

Bit	Symbol	Description	Reset value
11:0	APRPIO0_n	Edge select for start logic input PIO0_n (bit 0 = PIO0_0, ..., bit 11 = PIO0_11). 0 = Falling edge. 1 = Rising edge.	0
23:12	APRPIO1_n	Edge select for start logic input PIO1_n (bit 12 = PIO1_0, ..., bit 23 = PIO1_11). 0 = Falling edge. 1 = Rising edge.	0
31:24	APRPIO2_n	Edge select for start logic input PIO2_n (bit 24 = PIO2_0, ..., bit 31 = PIO2_7). 0 = Falling edge. 1 = Rising edge.	0

3.5.38 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 44.

Table 45. Start logic signal enable register 0 (STARTERP0, address 0x4004 8204) bit description

Bit	Symbol	Description	Reset value
11:0	ERPIO0_n	Enable start signal for start logic input PIO0_n (bit 0 = PIO0_0, ..., bit 11 = PIO0_11). 0 = Disabled. 1 = Enabled.	0
23:12	ERPIO1_n	Enable start signal for start logic input PIO1_n (bit 12 = PIO1_0, ..., bit 23 = PIO1_11). 0 = Disabled. 1 = Enabled.	0
31:24	ERPIO2_n	Enable start signal for start logic input PIO2_n (bit 24 = PIO2_0, ..., bit 31 = PIO2_7). 0 = Disabled. 1 = Enabled.	0

3.5.39 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 44. 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 46. Start logic reset register 0 (STARTRSRP0CLR, address 0x4004 8208) bit description

Bit	Symbol	Description	Reset value
11:0	RSRPIO0_n	Start signal reset for start logic input PIO0_n (bit 0 = PIO0_0, ..., bit 11 = PIO0_11). 0 = Do nothing. 1 = Write: reset start signal.	0
23:12	RSRPIO1_n	Start signal reset for start logic input PIO1_n (bit 12 = PIO1_0, ..., bit 23 = PIO1_11). 0 = Do nothing. 1 = Write: reset start signal.	0
31:24	RSRPIO2_n	Start signal reset for start logic input PIO2_n (bit 24 = PIO2_0, ..., bit 31 = PIO2_7). 0 = Do nothing. 1 = Write: reset start signal.	0

3.5.40 Start logic status register 0

This register reflects the status of the enabled start signal bits. The bit assignment is identical to Table 44. 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 47. Start logic status register 0 (STARTSRP0, address 0x4004 820C) bit description

Bit	Symbol	Description	Reset value
11:0	SRPIO0_n	Start signal status for start logic input PIO0_n (bit 0 = PIO0_0, ..., bit 11 = PIO0_11). 0 = No start signal received. 1 = Start signal pending.	0
23:12	SRPIO1_n	Start signal status for start logic input PIO1_n (bit 12 = PIO1_0, ..., bit 23 = PIO1_11). 0 = No start signal received. 1 = Start signal pending.	0
31:24	SRPIO2_n	Start signal status for start logic input PIO2_n (bit 24 = PIO2_0, ..., bit 31 = PIO2_7). 0 = No start signal received. 1 = Start signal pending.	0

3.5.41 Start logic edge control register 1

The STARTAPRP1 register controls the start logic inputs of ports 2 (PIO2_8 to PIO2_11) and 3 (PIO3_0 to PIO3_3). 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-up logic.

Every bit in the STARTAPRP1 register controls one port input and is connected to one wake-up interrupt in the NVIC. Bit 0 in the STARTAPRP1 register corresponds to interrupt 32, bit 1 to interrupt 33, up to bit 7 corresponding to interrupt 39 (see Table 66).

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 chip from Deep-sleep mode.

Table 48. Start logic edge control register 1 (STARTAPRP1, address 0x4004 8210) bit description

Bit	Symbol	Description	Reset value
3:0	APRPIO2_n	Edge select for start logic input PIO2_n (bit 0 = PIO2_8, ..., bit 3 = PIO2_11). 0 = Falling edge. 1 = Rising edge.	0
7:4	APRPIO3_n	Edge select for start logic input PIO3_n (bit 4 = PIO3_0, ..., bit 7 = PIO3_3). 0 = Falling edge. 1 = Rising edge.	0
31:8	-	Reserved	0

3.5.42 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 48.

Table 49. Start logic signal enable register 1 (STARTERP1, address 0x4004 8214) bit description

Bit	Symbol	Description	Reset value
3:0	ERPIO2_n	Enable start signal for start logic input PIO2_n (bit 0 = PIO2_8, ..., bit 3 = PIO2_11). 0 = Disabled. 1 = Enabled.	0
7:4	ERPIO3_n	Enable start signal for start logic input PIO3_n (bit 4 = PIO3_0, ..., bit 7 = PIO3_3). 0 = Disabled. 1 = Enabled.	0
31:8	-	Reserved	0

3.5.43 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 48. 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 50. Start logic reset register 1 (STARTRSRP1CLR, address 0x4004 8218) bit description

Bit	Symbol	Description	Reset value
3:0	RSRPIO2_n	Start signal reset for start logic input PIO2_n (bit 0 = PIO2_8, ..., bit 3 = PIO2_11). 0 = Do nothing. 1 = Write: reset start signal.	0
7:4	RSRPIO3_n	Start signal reset for start logic input PIO3_n (bit 4 = PIO3_0, ..., bit 7 = PIO3_3). 0 = Do nothing. 1 = Write: reset start signal.	0
31:8	-	Reserved	n/a

3.5.44 Start logic status register 1

This register reflects the status of the enabled start signals. The bit assignment is identical to Table 48.

Table 51. Start logic signal status register 1 (STARTSRP1, address 0x4004 821C) bit description

Bit	Symbol	Description	Reset value
3:0	SRPIO2_n	Start signal status for start logic input PIO2_n (bit 0 = PIO2_8, ..., bit 3 = PIO2_11). 0 = No start signal received. 1 = Start signal pending.	0
7:4	SRPIO3_n	Start signal status for start logic input PIO3_n (bit 4 = PIO3_0, ..., bit 7 = PIO3_3). 0 = No start signal received. 1 = Start signal pending.	0
31:8	-	Reserved	n/a

3.5.45 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.

This register **must be initialized at least once before entering Deep-sleep mode** with one of the four values shown in Table 52, depending on the part number.

Remark: Parts belonging to the LPC1300L series (LPC1311/01 and LPC1313/01) require different values in the PDSLEEPCFG register than parts belonging to the LPC1300 series.

Table 52. Allowed values for PDSLEEPCFG register

Configuration	WD oscillator on	WD oscillator off
BOD on	• LPC1300: PDSLEEPCFG = 0x0000 0FB7 • LPC1300L: PDSLEEPCFG = 0x0000 18B7	• LPC1300: PDSLEEPCFG = 0x0000 0FF7 • LPC1300L: PDSLEEPCFG = 0x0000 18F7
BOD off	• LPC1300: PDSLEEPCFG = 0x0000 0FBF • LPC1300L: PDSLEEPCFG = 0x0000 18BF	• LPC1300: PDSLEEPCFG = 0x0000 0FFF • LPC1300L: PDSLEEPCFG = 0x0000 18FF

Remark: Failure to initialize and program this register correctly may result in undefined behavior of the microcontroller. The values listed in Table 52 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 or a general purpose timer if they are needed for timing a wake-up event (see Section 3.10.3 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 15) and all peripheral clocks other than the timer clock must be disabled in the SYSAHBCLKCTRL register (see Table 25) before entering Deep-sleep mode.

  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.

Table 53. Deep-sleep configuration register (PDSLEEPCFG, address 0x4004 8230) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FIXEDVAL0	-	Reserved. **Always write these bits as 111.**	0
3	BOD_PD		BOD power-down control in Deep-sleep mode, see Table 52.	0
		0	Powered	
		1	Powered down	
5:4	FIXEDVAL1	-	Reserved. **Always write these bits as 11.**	0

Table 53. Deep-sleep configuration register (PDSLEEPCFG, address 0x4004 8230) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
6	WDTOSC_PD		Watchdog oscillator power control in Deep-sleep mode, see Table 52.	0
		0	Powered	
		1	Powered down	
12:7	FIXEDVAL2	-	Reserved. Bit value depends on part number: • For parts LPC1300 - **Always write these bits as 011111.** • For parts LPC1300L (parts LPC1311/01 and LPC1313/01) - **Always write these bits as 110001.**	0
31:13	-	0	Reserved	0

3.5.46 Wake-up configuration register

The bits in this register can be programmed to determine the state the chip must enter when it is waking up from 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.

Table 54. Wake-up configuration register (PDAWAKECFG, address 0x4004 8234) bit description

Bit	Symbol	Value	Description	Reset value
0	IRCOUT_PD		IRC oscillator output wake-up configuration	0
		0	Powered	
		1	Powered down	
1	IRC_PD		IRC oscillator power-down wake-up configuration	0
		0	Powered	
		1	Powered down	
2	FLASH_PD		Flash wake-up configuration	0
		0	Powered	
		1	Powered down	
3	BOD_PD		BOD wake-up configuration	0
		0	Powered	
		1	Powered down	
4	ADC_PD		ADC wake-up configuration	1
		0	Powered	
		1	Powered down	
5	SYSOSC_PD		System oscillator wake-up configuration	1
		0	Powered	
		1	Powered down	

Table 54. Wake-up configuration register (PDAWAKECFG, address 0x4004 8234) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
6	WDTOSC_PD		Watchdog oscillator wake-up configuration	1
		0	Powered	
		1	Powered down	
7	SYSPLL_PD		System PLL wake-up configuration	1
		0	Powered	
		1	Powered down	
8	USBPLL_PD		USB PLL wake-up configuration	1
		0	Powered	
		1	Powered down	
9	FIXEDVAL0	-	Reserved. **Always write this bit as 0.**	0
10	USBPAD_PD		USB pad wake-up configuration	1
		0	USB PHY powered	
		1	USB PHY powered down	
11	FIXEDVAL1	-	Reserved. **Always write this bit as 1.**	1
12	FIXEDVAL2	-	Reserved. **Always write this bit as 0.**	1
15:13	FIXEDVAL3	-	Reserved. **Always write this bit as 111.**	111
31:16	-	-	Reserved	0

3.5.47 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 chip is running, and a write will take effect immediately with the exception of the power-down signal to the IRC.

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

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

Table 55. 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	

Table 55. Power-down configuration register (PDRUNCFG, address 0x4004 8238) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
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]	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	USBPLL_PD		USB PLL power-down	1
		0	Powered	
		1	Powered down	
9	FIXEDVAL0	-	Reserved. **Always write this bit as 0.**	0
10	USBPAD_PD		USB pad power-down configuration	1
		0	USB PHY powered	
		1	USB PHY powered down (suspend mode)	
11	FIXEDVAL1	-	Reserved. **Always write this bit as 1.**	1
12	FIXEDVAL2	-	Reserved. **Always write this bit as 0.**	1
15:13	FIXEDVAL3	-	Reserved. **Always write this bit as 111.**	111
31:16	-	-	Reserved	0

[1] The system oscillator must be powered up and selected for the USB PLL to create a stable USB clock (see Table 20).

3.5.48 Device ID register

This device ID register is a read-only register and contains the device ID for each LPC13xx part. This register is also read by the ISP/IAP commands (see Section 21.13.11 and Section 21.13.11).

Table 56. Device ID register (DEVICE_ID, address 0x4004 83F4) bit description

Bit	Symbol	Description	Reset value
31:0	DEVICEID	Device ID for LPC13xx parts: 0x2C42 502B = LPC1311FHN33 0x2C40 102B = LPC1313FHN33 0x2C40 102B = LPC1313FBD48 0x3D01 402B = LPC1342FHN33 0x3D01 402B = LPC1342FBD48 0x3D00 002B = LPC1343FHN33 0x3D00 002B = LPC1343FBD48 0x1816 902B = LPC1311FHN33/01 0x1830 102B = LPC1313FHN33/01 0x1830 102B = LPC1313FBD48/01	part-dependent

3.6 Reset

Reset has four sources on the LPC13xx: 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 chip Reset by any source, once the operating voltage attains a usable level, starts the IRC causing reset to remain asserted until the external Reset is de-asserted, the oscillator is running, and the flash controller has completed its initialization.

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

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

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

3.7 Start-up behavior

See Figure 4 for the start-up timing after reset. The IRC is the default clock at Reset and provides a clean system clock shortly after the supply voltage reaches the threshold value of 1.8 V.

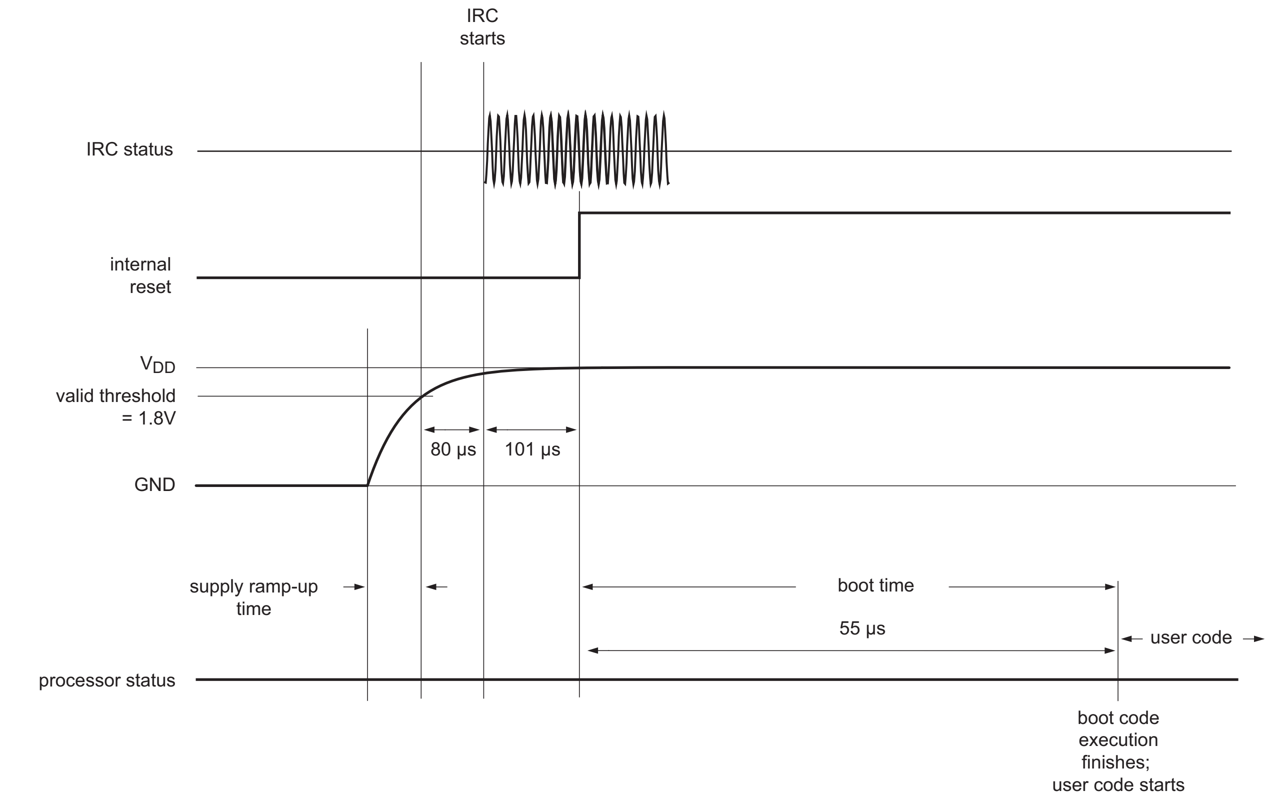


Fig 4. Start-up timing

3.8 Brown-out detection

The LPC13xx includes four levels for monitoring the voltage on the V_{DD} pin. If this voltage falls below one of the four selected levels, the BOD asserts an interrupt signal to the NVIC. This signal can be enabled for interrupt in the Interrupt Enable Register in the NVIC in order to cause a CPU interrupt; if not, software can monitor the signal by reading a dedicated status register. One additional threshold level can be selected to cause a forced reset of the chip on the LPC1311/13/42/43 parts. Four additional threshold levels for forced reset can be selected on the LPC1311/01 and LPC1313/01 parts.

3.9 Power management

The LPC13xx support a variety of power control features. In Active mode, when the chip is running, power and clocks to selected peripherals can be optimized for power consumption. In addition, there are 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.

3.9.1 Active mode

In Active mode, the ARM Cortex-M3 core and memories are clocked by the system clock, and peripherals are clocked by the system clock or a dedicated peripheral clock.

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

3.9.1.1 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 25).
- The power to various analog blocks (USB, PLL, oscillators, the ADC, the BOD circuit, and the flash block) can be controlled at any time individually through the PDRUNCFG register (Table 55).
- 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 24).
- 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 26 to Table 36).

3.9.2 Sleep mode

In Sleep mode, the system clock to the ARM Cortex-M3 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.

3.9.2.1 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.

3.9.2.2 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 61).
2. The SLEEPDEEP bit in the ARM Cortex-M3 SCR register must be set to zero.

3. Use the ARM Cortex-M3 Wait-For-Interrupt (WFI) instruction.

3.9.2.3 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.

3.9.3 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.

Deep-sleep mode eliminates all power used by the flash and 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.

3.9.3.1 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 53) register:

- 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 timer-controlled wake-up (see Section 3.10.3). 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 15).
- 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 or one of the general-purpose timers should be enabled in SYSAHBCLKCTRL register to minimize power consumption.

3.9.3.2 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 61).
2. Select the power configuration in Deep-sleep mode in the PDSLEEPCFG (Table 53) register.
    a. If a timer-controlled wake-up is needed, ensure that the watchdog oscillator is powered in the PDRUNCFG register and switch the clock source to WD oscillator in the MAINCLKSEL register (Table 22).
    b. If timer-controlled wake-up is not needed and 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 22). This ensures that the system clock is shut down glitch-free.

3. Select the power configuration after wake-up in the PDAWAKECFG (Table 54) register.
4. If an external pin is used for wake-up, enable and clear the wake-up pin in the start logic registers (Table 44 to Table 51), and enable the start logic interrupt in the NVIC.
5. In the SYSAHBCLKCTRL register (Table 25), disable all peripherals except counter/timer or WDT if needed.
6. Write one to the SLEEPDEEP bit in the ARM Cortex-M3 SCR register.
7. Use the ARM WFI instruction.

3.9.3.3 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 and PIO1_0 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.
- Input signal to the start logic created by a match event on one of the general purpose timer external match outputs. The pin holding the timer match function must be enabled as start logic input in the NVIC, the corresponding timer must be enabled in the SYSAHBCLKCTRL register, and the watchdog oscillator must be running in Deep-sleep mode (for details see Section 3.10.3).
- 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 42).
- 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.
- A reset signal from the 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 IRC.

3.9.4 Deep power-down mode

In Deep power-down mode, power and clocks are shut off to the entire chip with the exception of the WAKEUP pin. The Deep power-down mode is controlled by the PMU (see Chapter 4).

During Deep power-down mode, the contents of the SRAM and registers are not retained except for a small amount of data which can be stored in the five 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.

3.9.4.1 Power configuration in Deep power-down mode

Deep power-down mode has no configuration options. All clocks, the core, and all peripherals are powered down. Only the WAKEUP pin is powered.

3.9.4.2 Programming Deep power-down mode

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 61).
2. Store data to be retained in the general purpose registers (Table 62).
3. Write one to the SLEEPDEEP bit in the ARM Cortex-M3 SCR register.
4. For the LPC1311/13/42/43, 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. This step is not required for the LPC1311/01 and LPC1313/01.
5. Use the ARM WFI instruction.

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

3.9.4.3 Wake-up from Deep power-down mode

Pulling the WAKEUP pin LOW wakes up the LPC13xx from Deep power-down, and the chip goes through the entire reset process (Section 3.6). The minimum pulse width for the HIGH-to-LOW transition on the WAKEUP pin is 50 ns.

1. A wake-up signal is generated when a HIGH-to-LOW transition occurs 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 chip re-boots.
   - All registers except the GPREG0 to GPREG4 will be in their reset state.
2. Once the chip has booted, read the deep power-down flag in the PCON register (Table 61) to verify that the reset was caused by a wake-up event from Deep power-down.
3. Clear the deep power-down flag in the PCON register (Table 61).
4. (Optional) Read the stored data in the general purpose registers (Table 62 and Table 63).
5. Set up the PMU for the next Deep power-down cycle.

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

3.10 Deep-sleep mode details

3.10.1 IRC oscillator

The IRC is the only oscillator on the LPC13xx that can always shut down glitch-free. Therefore it is recommended that the user switches the clock source to IRC before the chip enters Deep-sleep mode.

3.10.2 Start logic

The Deep-sleep mode is exited when the start logic indicates an interrupt to the ARM core. The various port pins (see Table 6) are connected to the start logic and serve as wake-up pins. The user must program the start logic 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 39 in the NVIC correspond to 40 PIO pins (see Section 3.5.37 and Section 3.5.41).

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 when enabled. Therefore, the start logic signals should be reset (see Table 46 and Table 50) before use.

The start logic can also be used in Active mode to provide a vectored interrupt using the LPC13xx's input pins.

3.10.3 Using the general purpose counter/timers to create a self-wake-up event

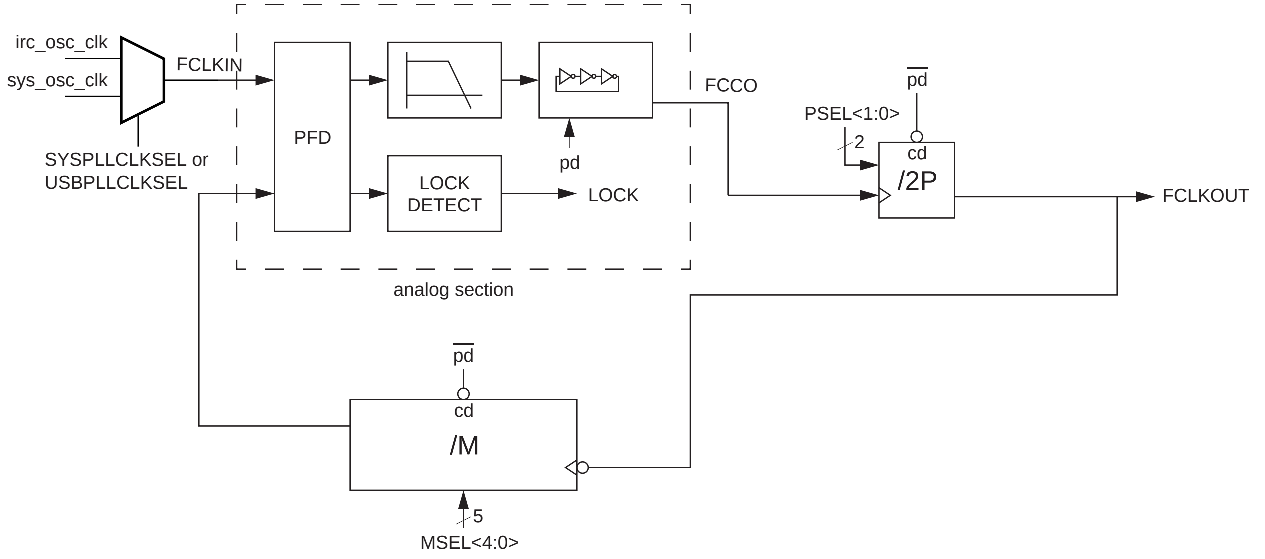
If enabled in Deep-sleep mode through the SYSAHBCLKCFG register, the counter/timers can count clock cycles of the watchdog oscillator and create a match event when the number of cycles equals a preset match value. The match event causes the corresponding match output pin to go HIGH, LOW, or toggle. The state of the match output pin is also monitored by the start logic and can trigger a wake-up interrupt if that pin is enabled in the NVIC and the start logic trigger is configured accordingly in the start logic edge control register (see Table 44 and Table 48).

The following steps must be performed to configure the counter/timer and create a timed Deep-sleep self-wake-up event:

1. Configure the port pin as match output in the IOCONFIG block. All pins with a match function are also inputs to the start logic.
2. In the corresponding counter/timer, set the match value, and configure the match output for the selected pin.
3. Select the watchdog oscillator to run in Deep-sleep mode in the PDSLEEPCFG register.
4. Switch the clock source to the watchdog oscillator in the MAINCLKSEL register (Table 22) and ensure the watchdog oscillator is powered in the PDRUNCFG register.
5. Enable the pin, configure its edge detect function, and reset the start logic in the start logic registers (Table 46 to Table 50), and enable the interrupt in the NVIC.
6. Disable all other peripherals in the SYSAHBCLKCTRL register.
7. Ensure that the DPDEN bit in the PCON register is set to zero (Table 61).
8. Write one to the SLEEPDEEP bit in the ARM Cortex-M3 SCR register.
9. Start the counter/timer.
10. Use the ARM WFI instruction to enter Deep-sleep mode.

3.11 PLL (System PLL and USB PLL) functional description

The LPC13xx uses the system PLL to create the clocks for the core and peripherals. On the LPC134x parts, there is a second, identical PLL to create the USB clock.



(1) Not on USB PLL.

Fig 5. System and USB PLL block diagram

The block diagram of this PLL is shown in Figure 5. 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.

3.11.1 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.

3.11.2 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 (or USBPLL_PD) bits to one in the Power-down configuration register (Table 55). 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 (or USBPLL_PD) bits 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.

3.11.3 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 and Table 12. 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 and Table 12.

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.

3.11.4 Frequency selection

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

Table 57. PLL frequency parameters

Parameter	System PLL	USB PLL
FCLKIN	Frequency of sys_pllclkin (input clock to the system PLL) from the SYSPLLCLKSEL multiplexer (see Section 3.5.11).	Frequency of usb_pllclkin (input clock to the USB PLL) from the USBPLLCLKSEL multiplexer (see Section 3.5.24).
FCCO	Frequency of the Current Controlled Oscillator (CCO); 156 to 320 MHz.	Frequency of the Current Controlled Oscillator (CCO); 156 to 320 MHz.

Table 57. PLL frequency parameters

Parameter	System PLL	USB PLL
FCLKOUT	Frequency of sys_pllclkout; < 100 MHz	Frequency of usb_pllclkout; < 100 MHz
P	System PLL post divider ratio; PSEL bits in SYSPLLCTRL (see Section 3.5.3).	USB PLL post divider ratio; PSEL bits in USBPLLCTRL (see Section 3.5.5).
M	System PLL feedback divider register; MSEL bits in SYSPLLCTRL (see Section 3.5.3).	USB PLL feedback divider register; MSEL bits in USBPLLCTRL (see Section 3.5.5).

3.11.4.1 Normal mode

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

$$FCLKOUT = M \times FCLKIN = (FCCO)/(2 \times P) \tag{1}$$

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

1. Specify the input clock frequency Fclkin.
2. Calculate M to obtain the desired output frequency FCLKOUT with M = FCLKOUT / FCLKIN.
3. Find a value so that FCCO = 2 × P × FCLKOUT.
4. Verify that all frequencies and divider values conform to the limits specified in Table 10 and Table 12.
5. Ensure that FCLKOUT < 100 MHz.

Table 58 shows how to configure the PLL for a 12 MHz crystal oscillator using the SYSPLLCTRL or USBPLLCTRL registers (Table 10 or Table 11). The main clock is equivalent to the system clock if the system clock divider SYSAHBCLKDIV is set to one (see Table 24).

Table 58. PLL configuration examples

PLL input clock sys_pllclkin (FCLKIN)	Main clock (FCLKOUT)	MSEL bits Table 10 (binary)	M divider value	PSEL bits Table 10 (binary)	P divider value	FCCO frequency
12 MHz	72 MHz	00101	6	01	2	288 MHz
12 MHz	48 MHz	00011	4	01	2	192 MHz
12 MHz	36 MHz	00010	3	10	4	288 MHz
12 MHz	24 MHz	00001	2	10	4	192 MHz

3.11.4.2 Power-down mode

In this mode, the internal current reference will be turned off, the oscillator and the phase-frequency detector will be stopped and the dividers will enter a reset state. While in 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 SYSPLL_PD (or USBPLL_PD) bits to zero in the Power-down configuration register (Table 55), the PLL will resume its normal operation and will make the lock signal high once it has regained lock on the input clock.

3.12 Flash memory access

Depending on the system clock frequency, access to the flash memory can be configured with various access times by writing to the FLASHCFG register at address 0x4003 C010. This register is part of the flash controller block (see Figure 2).

Remark: Improper setting of this register may result in incorrect operation of the LPC13xx flash memory.

Table 59. Flash configuration register (FLASHCFG, address 0x4003 C010) bit description

Bit	Symbol	Value	Description	Reset value
1:0	FLASHTIM		Flash memory access time. FLASHTIM +1 is equal to the number of system clocks used for flash access.	10
		0x0	1 system clock flash access time (for system clock frequencies of up to 20 MHz).	
		0x1	2 system clocks flash access time (for system clock frequencies of up to 40 MHz).	
		0x2	3 system clocks flash access time (for system clock frequencies of up to 72 MHz).	
		0x3	Reserved.	
31:2	-	-	Reserved. **User software must not change the value of these bits. Bits 31:2 must be written back exactly as read**.	-

UM10375

Chapter 4: LPC13xx Power Management Unit (PMU)

Rev. 5 — 21 June 2012 User manual

4.1 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.

4.2 Register description

Table 60. 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
GPREG4	R/W	0x014	General purpose register 4	0x0

4.2.1 Power control register

The power control register selects whether one of the ARM Cortex-M3 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 3.9 for details on how to enter the power-down modes.

Table 61. 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-M3 core turned off).	
		1	ARM WFI will enter Deep-power down mode (ARM Cortex-M3 core powered-down).	
7:2	-	-	Reserved. Do not write ones to this bit.	0x0
8	SLEEPFLAG		Sleep mode flag	0
		0	Read: No power-down mode entered. LPC13xx 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

Table 61. Power control register (PCON, address 0x4003 8000) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
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

4.2.2 General purpose registers 0 to 3

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

Table 62. General purpose registers 0 to 3 (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

4.2.3 General purpose register 4

The general purpose register 4 retains data through the Deep power-down mode when power is still applied to the V_{DD} pin but the chip has entered Deep power-down mode. Only a “cold” boot, when all power has been completely removed from the chip, will reset the general purpose registers.

The hysteresis of the WAKEUP pin in Deep power-down mode can be controlled by bit 10 of this register.

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

Table 63. General purpose register 4 (GPREG4, 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.	
31:11	GPDATA		Data retained during Deep power-down mode.	0x0

4.3 Functional description

See Section 3.9 for details on power management and the Deep power-down mode.

UM10375

Chapter 5: LPC13xx Power profiles

Rev. 5 — 21 June 2012 **User manual**

5.1 How to read this chapter

The power profiles are available for parts LPC1311/01 and LPC1313/01 only (LPC1300L series).

5.2 Features

- Includes ROM-based application services
- Power Management services
- Clocking services

5.3 Description

The API calls to the ROM are performed by executing functions which are pointed by a pointer within the ROM Driver Table. Figure 6 shows the pointer structure used to call the Power Profiles API.

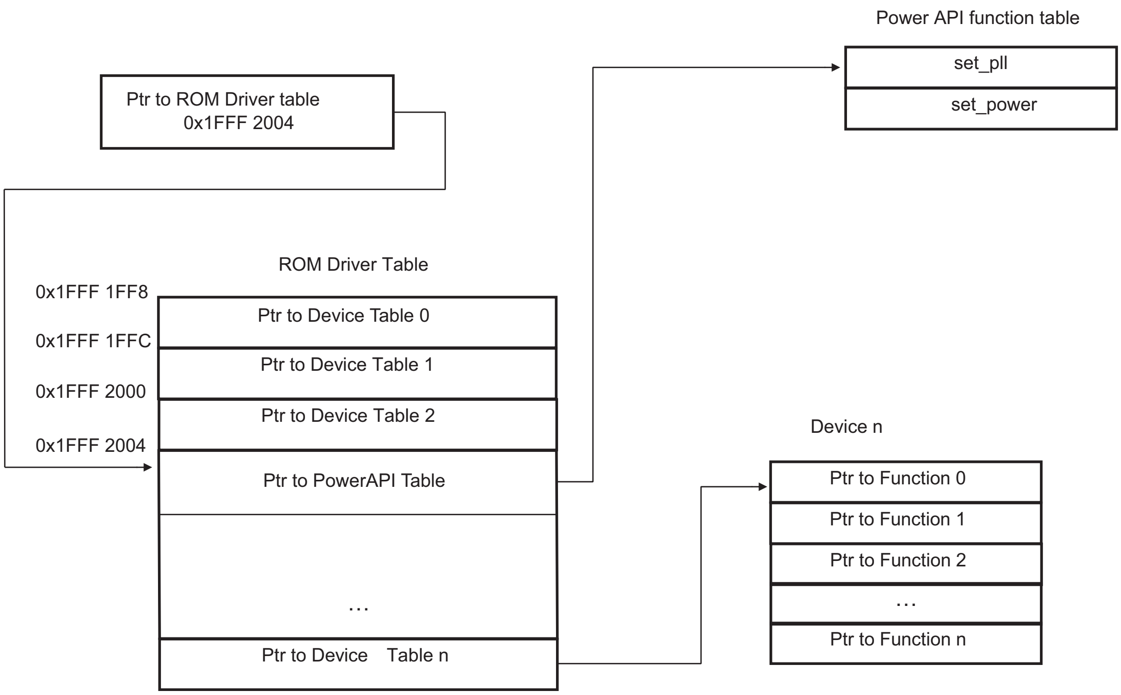


Fig 6. Power profiles pointer structure

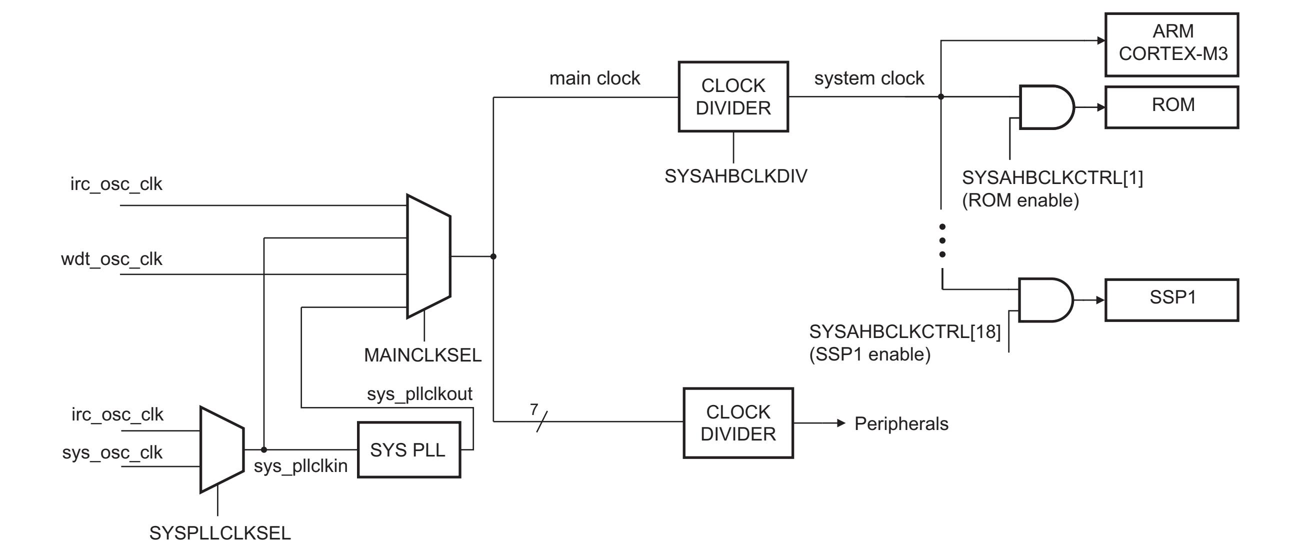


Fig 7. LPC1311/01 and LPC1313/01 clock configuration for power API use

5.4 Definitions

The following elements have to be defined in an application that uses the power profiles:

```
typedef struct _PWRD {
     void (*set_pll)(unsigned int cmd[], unsigned int resp[]);
     void (*set_power)(unsigned int cmd[], unsigned int resp[]);
} PWRD;
typedef struct _ROM {
     const PWRD * pWRD;
} ROM;
ROM ** rom = (ROM **) (0x1FFF1FF8 + 3 * (sizeof(ROM**));
unsigned int command[4], result[2];
```

5.5 Clocking routine

5.5.1 set_pll

This routine sets up the system PLL according to the calling arguments. If the expected clock can be obtained by simply dividing the system PLL input, *set_pll* bypasses the PLL to lower system power consumption.

Remark: Before this routine is invoked, the PLL clock source (IRC/system oscillator) must be selected (Table 18), the main clock source must be set to the input clock to the system PLL (Table 22) and the system/AHB clock divider must be set to 1 (Table 24).

set_pll attempts to find a PLL setup that matches the calling parameters. Once a combination of a feedback divider value (SYSPLLCTRL, M), a post divider ratio (SYSPLLCTRL, P) and the system/AHB clock divider (SYSAHBCLKDIV) is found, *set_pll* applies the selected values and switches the main clock source selection to the system PLL clock out (if necessary).

The routine returns a result code that indicates if the system PLL was successfully set (PLL_CMD_SUCCESS) or not (in which case the result code identifies what went wrong). The current system frequency value is also returned. The application should use this information to adjust other clocks in the device (the SSP, UART, and WDT clocks, and/or clockout).

Table 64. set_pll routine

Routine	set_pll
Input	**Param0:** system PLL input frequency (in kHz) **Param1:** expected system clock (in kHz) **Param2:** mode (CPU_FREQ_EQU, CPU_FREQ_LTE, CPU_FREQ_GTE, CPU_FREQ_APPROX) **Param3:** system PLL lock time-out
Result	**Result0:** PLL_CMD_SUCCESS | PLL_INVALID_FREQ | PLL_INVALID_MODE | PLL_FREQ_NOT_FOUND | PLL_NOT_LOCKED **Result1:** system clock (in kHz)

The following definitions are needed when making set_pll power routine calls:

```
/* set_pll mode options */
#define    CPU_FREQ_EQU         0
#define    CPU_FREQ_LTE         1
#define    CPU_FREQ_GTE         2
#define    CPU_FREQ_APPROX      3
/* set_pll result0 options */
#define    PLL_CMD_SUCCESS      0
#define    PLL_INVALID_FREQ     1
#define    PLL_INVALID_MODE     2
#define    PLL_FREQ_NOT_FOUND   3
#define    PLL_NOT_LOCKED       4
```

For a simplified clock configuration scheme see Figure 7. For more details see Figure 3.

5.5.1.1 Param0: system PLL input frequency and Param1: expected system clock

set_pll looks for a setup in which the system PLL clock does not exceed 72 MHz. It easily finds a solution when the ratio between the expected system clock and the system PLL input frequency is an integer value, but it can also find solutions in other cases.

The system PLL input frequency (*Param0*) must be between 10000 to 25000 kHz (10 MHz to 25 MHz) inclusive. The expected system clock (*Param1*) must be between 1 and 72000 kHz inclusive. If either of these requirements is not met, *set_pll* returns PLL_INVALID_FREQ and returns *Param0* as *Result1* since the PLL setting is unchanged.

5.5.1.2 Param2: mode

The first priority of *set_pll* is to find a setup that generates the system clock at exactly the rate specified in *Param1*. If it is unlikely that an exact match can be found, input parameter mode (*Param2*) should be used to specify if the actual system clock can be less than or equal, greater than or equal or approximately the value specified as the expected system clock (*Param1*).

A call specifying CPU_FREQ_EQU will only succeed if the PLL can output exactly the frequency requested in *Param1*.

CPU_FREQ_LTE can be used if the requested frequency should not be exceeded (such as overall current consumption and/or power budget reasons).

CPU_FREQ_GTE helps applications that need a minimum level of CPU processing capabilities.

CPU_FREQ_APPROX results in a system clock that is as close as possible to the requested value (it may be greater than or less than the requested value).

If an illegal mode is specified, *set_pll* returns PLL_INVALID_MODE. If the expected system clock is out of the range supported by this routine, *set_pll* returns PLL_FREQ_NOT_FOUND. In these cases the current PLL setting is not changed and *Param0* is returned as *Result1*.

5.5.1.3 Param3: system PLL lock time-out

It should take no more than 100 μs for the system PLL to lock if a valid configuration is selected. If *Param3* is zero, *set_pll* will wait indefinitely for the PLL to lock. A non-zero value indicates how many times the code will check for a successful PLL lock event before it returns PLL_NOT_LOCKED. In this case the PLL settings are unchanged and *Param0* is returned as *Result1*.

Remark: The time it takes the PLL to lock depends on the selected PLL input clock source (IRC/system oscillator) and its characteristics. The selected source can experience more or less jitter depending on the operating conditions such as power supply and/or ambient temperature. This is why it is suggested that when a good known clock source is used and a PLL_NOT_LOCKED response is received, the set_pll routine should be invoked several times before declaring the selected PLL clock source invalid.

Hint: setting *Param3* equal to the system PLL frequency [Hz] divided by 10000 will provide more than enough PLL lock-polling cycles.

5.5.1.4 Code examples

The following examples illustrate some of the features of *set_pll* discussed above.

5.5.1.4.1 Invalid frequency (device maximum clock rate exceeded)

```
command[0] = 12000;
command[1] = 84000;
command[2] = CPU_FREQ_EQU;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock and a system clock of exactly 84 MHz. The application was ready to infinitely wait for the PLL to lock. But the expected system clock of 84 MHz exceeds the maximum of 72 MHz. Therefore *set_pll* returns PLL_INVALID_FREQ in *result[0]* and 12000 in *result[1]* without changing the PLL settings.

5.5.1.4.2 Invalid frequency selection (system clock divider restrictions)

```
command[0] = 12000;
command[1] = 40;
command[2] = CPU_FREQ_LTE;
```

```
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock, a system clock of no more than 40 kHz and no time-out while waiting for the PLL to lock. Since the maximum divider value for the system clock is 255 and running at 40 kHz would need a divide by value of 300, *set_pll* returns PLL_INVALID_FREQ in *result[0]* and 12000 in *result[1]* without changing the PLL settings.

5.5.1.4.3 Exact solution cannot be found (PLL)

```
command[0] = 12000;
command[1] = 25000;
command[2] = CPU_FREQ_EQU;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock and a system clock of exactly 25 MHz. The application was ready to infinitely wait for the PLL to lock. Since there is no valid PLL setup within earlier mentioned restrictions, *set_pll* returns PLL_FREQ_NOT_FOUND in *result[0]* and 12000 in *result[1]* without changing the PLL settings.

5.5.1.4.4 System clock less than or equal to the expected value

```
command[0] = 12000;
command[1] = 25000;
command[2] = CPU_FREQ_LTE;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock, a system clock of no more than 25 MHz and no locking time-out. *set_pll* returns PLL_CMD_SUCCESS in *result[0]* and 24000 in *result[1]*. The new system clock is 24 MHz.

5.5.1.4.5 System clock greater than or equal to the expected value

```
command[0] = 12000;
command[1] = 25000;
command[2] = CPU_FREQ_GTE;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock, a system clock of at least 25 MHz and no locking time-out. *set_pll* returns PLL_CMD_SUCCESS in *result[0]* and 36000 in *result[1]*. The new system clock is 36 MHz.

5.5.1.4.6 System clock approximately equal to the expected value

```
command[0] = 12000;
command[1] = 16500;
command[2] = CPU_FREQ_APPROX;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock, a system clock of approximately 16.5 MHz and no locking time-out. *set_pll* returns PLL_CMD_SUCCESS in *result[0]* and 16000 in *result[1]*. The new system clock is 16 MHz.

5.6 Power routine

5.6.1 set_power

This routine configures the device's internal power control settings according to the calling arguments. The goal is to reduce active power consumption while maintaining the feature of interest to the application close to its optimum.

Remark: The set_power routine was designed for systems employing the configuration of SYSAHBCLKDIV = 1 (System clock divider register, see Table 24 and Figure 7). Using this routine in an application with the system clock divider not equal to 1 might not improve microcontroller's performance as much as in setups when the main clock and the system clock are running at the same rate.

set_power returns a result code that reports whether the power setting was successfully changed or not.

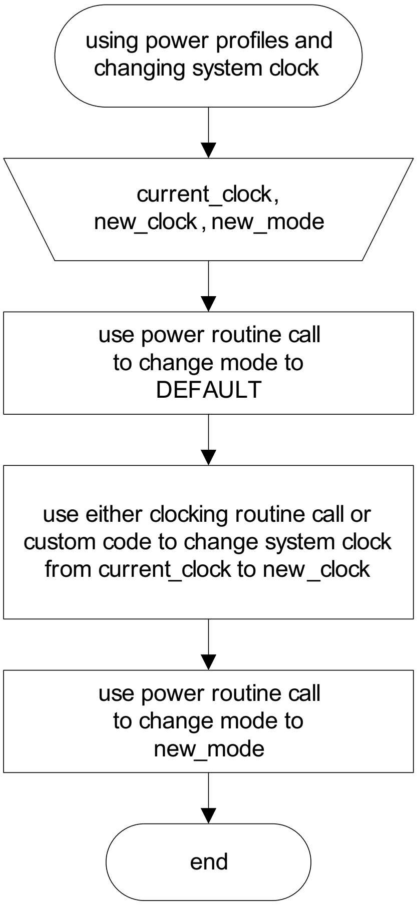

using power profiles and changing system clock
current_clock,
new_clock, new_mode
use power routine call to change mode to DEFAULT
use either clocking routine call or custom code to change system clock from current_clock to new_clock
use power routine call to change mode to new_mode
end

Fig 8. Power profiles usage

Table 65. set_power routine

Routine	set_power
Input	**Param0:** main clock (in MHz) **Param1:** mode (PWR_DEFAULT, PWR_CPU_PERFORMANCE, PWR_EFFICIENCY, PWR_LOW_CURRENT) **Param2:** system clock (in MHz)
Result	**Result0:** PWR_CMD_SUCCESS | PWR_INVALID_FREQ | PWR_INVALID_MODE

The following definitions are needed for set_power routine calls:

```
/* set_power mode options */
#define    PWR_DEFAULT          0
#define    PWR_CPU_PERFORMANCE  1
#define    PWR_EFFICIENCY       2
#define    PWR_LOW_CURRENT      3
/* set_power result0 options */
#define    PWR_CMD_SUCCESS      0
#define    PWR_INVALID_FREQ     1
#define    PWR_INVALID_MODE     2
```

For a simplified clock configuration scheme see Figure 7. For more details see Figure 3.

5.6.1.1 Param0: main clock

The main clock is the clock rate the microcontroller uses to source the system's and the peripherals' clock. It is configured by either a successful execution of the clocking routine call or a similar code provided by the user. This operand must be an integer between 1 to 72 MHz inclusive. If a value out of this range is supplied, *set_power* returns PWR_INVALID_FREQ and does not change the power control system.

5.6.1.2 Param1: mode

The input parameter mode (*Param1*) specifies one of four available power settings. If an illegal selection is provided, *set_power* returns PWR_INVALID_MODE and does not change the power control system.

PWR_DEFAULT keeps the device in a baseline power setting similar to its reset state.

PWR_CPU_PERFORMANCE configures the microcontroller so that it can provide more processing capability to the application. CPU performance is 30% better than the default option.

PWR_EFFICIENCY setting was designed to find a balance between active current and the CPU's ability to execute code and process data. In this mode the device outperforms the default mode both in terms of providing higher CPU performance and lowering active current.

PWR_LOW_CURRENT is intended for those solutions that focus on lowering power consumption rather than CPU performance.

5.6.1.3 Param2: system clock

The system clock is the clock rate at which the microcontroller core is running when *set_power* is called. This parameter is an integer between from 1 and 72 MHz inclusive.

5.6.1.4 Code examples

The following examples illustrate some of the *set_power* features discussed above.

5.6.1.4.1 Invalid frequency (device maximum clock rate exceeded)

```
command[0] = 75;
command[1] = PWR_CPU_PERFORMANCE;
command[2] = 75;
(*rom)->pWRD->set_power(command, result);
```

The above setup would be used in a system running at the main and system clock of 75 MHz, with a need for maximum CPU processing power. Since the specified 75 MHz clock is above the 72 MHz maximum, *set_power* returns PWR_INVALID_FREQ in *result[0]* without changing anything in the existing power setup.

5.6.1.4.2 An applicable power setup

```
command[0] = 24;
command[1] = PWR_CPU_EFFICIENCY;
command[2] = 24;
(*rom)->pWRD->set_power(command, result);
```

The above code specifies that an application is running at the main and system clock of 24 MHz with emphasis on efficiency. *set_power* returns PWR_CMD_SUCCESS in *result[0]* after configuring the microcontroller’s internal power control features.

UM10375

Chapter 6: LPC13xx Interrupt controller

Rev. 5 — 21 June 2012 User manual

6.1 How to read this chapter

Interrupts 47 and 48 in Table 66 are available on parts LPC1342/43 with USB only. These interrupts are reserved on parts LPC1311/13.

Interrupt 57 is available for part LPC1313FBD48/01 only (48-pin package, LPC1300L series). This interrupt is reserved on parts LPC1311/13/42/43 and LPC1311FHN33/01 and LPC1313FHN33/01.

The implementation of start logic wake-up interrupts depends on how many PIO port pins are available (see Section 3.1). For HVQFN packages only wake-up interrupts 0 to 24 and interrupt 38 are available.

6.2 Introduction

The Nested Vectored Interrupt Controller (NVIC) is an integral part of the Cortex-M3. The tight coupling to the CPU allows for low interrupt latency and efficient processing of late arriving interrupts.

Refer to the *Cortex-M3 Technical Reference Manual* for details of NVIC operation.

6.3 Features

- Nested Vectored Interrupt Controller that is an integral part of the ARM Cortex-M3.
- Tightly coupled interrupt controller provides low interrupt latency.
- Controls system exceptions and peripheral interrupts.
- In the LPC13xx, the NVIC supports up to 56 vectored interrupts.
- 8 programmable interrupt priority levels with hardware priority level masking.
- Relocatable vector table.
- Software interrupt generation.

6.4 Interrupt sources

Table 66 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 66. Connection of interrupt sources to the Vectored Interrupt Controller

Exception Number	Vector Offset	Function	Flag(s)
39 to 0		start logic wake-up interrupts	Each interrupt is connected to a PIO input pin serving as wake-up pin from Deep-sleep mode (see Section 3.5.37 and Section 3.5.41). Interrupts 0 to 11 are connected to PIO0_0 to PIO0_11; interrupts 12 to 23 are connected to PIO1_0 to PIO1_11; interrupts 24 to 35 are connected to PIO2_0 to PIO2_11; interrupts 36 to 39 are connected to PIO3_0 to PIO3_3.[1]
40	0xA0	I2C0	SI (state change)
41	0xA4	CT16B0	Match 0 - 2 Capture 0
42	0xA8	CT16B1	Match 0 - 1 Capture 0
43	0xAC	CT32B0	Match 0 - 3 Capture 0
44	0xB0	CT32B1	Match 0 - 3 Capture 0
45	0xB4	SSP0	Tx FIFO half empty Rx FIFO half full Rx Timeout Rx Overrun
46	0xB8	UART	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Control Change End of Auto-Baud (ABEO) Auto-Baud Time-Out (ABTO)
47	0xBC	USB IRQ interrupt	USB low-priority interrupt
48	0xC0	USB FIQ interrupt	USB high-priority interrupt
49	0xC4	ADC	A/D Converter end of conversion
50	0xC8	WDT	Watchdog interrupt (WDINT)
51	0xCC	BOD	Brown-out detect
52	-	-	Reserved
53	0xD4	PIO_3	GPIO interrupt status of port 3
54	0xD8	PIO_2	GPIO interrupt status of port 2
55	0xDC	PIO_1	GPIO interrupt status of port 1
56	0xE0	PIO_0	GPIO interrupt status of port 0
57	0xE4	SSP1	Tx FIFO half empty Rx FIFO half full Rx Timeout Rx Overrun

[1] See Section 3.1 for wake-up pins not used in the HVQFN package.

6.5 Vector table remapping

The Cortex-M3 incorporates a mechanism that allows remapping the interrupt vector table to alternate locations in the memory map. This is controlled via the Vector Table Offset Register (VTOR) contained in the Cortex-M3.

The vector table may be located anywhere within the bottom 1 GB of Cortex-M3 address space. The vector table should be located on a 256 word (1024 byte) boundary to insure alignment on LPC13xx family devices. Refer to the ARM Cortex-M3 User Guide for details of the Vector Table Offset feature.

ARM describes bit 29 of the VTOR (TBLOFF) as selecting a memory region, either code or SRAM. For simplicity, this bit can be thought as simply part of the address offset since the split between the “code” space and the “SRAM” space occurs at the location corresponding to bit 29 in a memory address.

Example:

To place the vector table at the beginning of the static RAM, starting at address 0x1000 0000, place the value 0x1000 0000 in the VTOR register. This indicates address 0x1000 0000 in the code space, since bit 29 of the VTOR equals 0.

6.6 Register description

The following table summarizes the registers in the NVIC as implemented in the LPC13xx. The Cortex-M3 User Guide provides a functional description of the NVIC.

Table 67. Register overview: NVIC (base address 0xE000 E000)

Name	Access	Address offset	Description	Reset value
ISER0	RW	0x100	Interrupt Set-Enable Register 0. This register allows enabling interrupts and reading back the interrupt enables for specific peripheral functions.	0
ISER1	RW	0x104	Interrupt Set-Enable Register 1. This register allows enabling interrupts and reading back the interrupt enables for specific peripheral functions.	0
ICER0	RW	0x180	Interrupt Clear-Enable Register 0. This register allows disabling interrupts and reading back the interrupt enables for specific peripheral functions.	0
ICER1	RW	0x184	Interrupt Clear-Enable Register 1. This register allows disabling interrupts and reading back the interrupt enables for specific peripheral functions.	0
ISPR0	RW	0x200	Interrupt Set-Pending Register 0. This register allows changing the interrupt state to pending and reading back the interrupt pending state for specific peripheral functions.	0
ISPR1	RW	0x204	Interrupt Set-Pending Register 1. This register allows changing the interrupt state to pending and reading back the interrupt pending state for specific peripheral functions.	0
ICPR0	RW	0x280	Interrupt Clear-Pending Register 0. This register allows changing the interrupt state to not pending and reading back the interrupt pending state for specific peripheral functions.	0
ICPR1	RW	0x284	Interrupt Clear-Pending Register 1. This register allows changing the interrupt state to not pending and reading back the interrupt pending state for specific peripheral functions.	0
IABR0	RO	0x300	Interrupt Active Bit Register 0. This register allows reading the current interrupt active state for specific peripheral functions.	0
IABR1	RO	0x304	Interrupt Active Bit Register 1. This register allows reading the current interrupt active state for specific peripheral functions.	0
IPR0	RW	0x400	Interrupt Priority Registers 0. This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0
IPR1	RW	0x404	Interrupt Priority Registers 1 This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0
IPR2	RW	0x408	Interrupt Priority Registers 2. This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0
IPR3	RW	0x40C	Interrupt Priority Registers 3. This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0
IPR4	RW	0x410	Interrupt Priority Registers 4. This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0
IPR5	RW	0x414	Interrupt Priority Registers 5. This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0
IPR6	RW	0x418	Interrupt Priority Registers 6. This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0
IPR7	RW	0x41C	Interrupt Priority Registers 7. This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0
IPR8	RW	0x420	Interrupt Priority Registers 8 This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0

Table 67. Register overview: NVIC (base address 0xE000 E000) *…continued*

Name	Access	Address offset	Description	Reset value
IPR9	RW	0x424	Interrupt Priority Registers 9 This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0
IPR10	RW	0x428	Interrupt Priority Registers 10 This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0
IPR11	RW	0x42C	Interrupt Priority Registers 11 This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0
IPR12	RW	0x430	Interrupt Priority Registers 12 This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0
IPR13	RW	0x434	Interrupt Priority Registers 13 This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0
IPR14	RW	0x438	Interrupt Priority Registers 14 This register allows assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	0
STIR	WO	0xF00	Software Trigger Interrupt Register. This register allows software to generate an interrupt.	0

6.6.1 Interrupt Set-Enable Register 0 register

The ISER0 register allows enabling the first 32 peripheral interrupts, or for reading the enabled state of those interrupts. The remaining interrupts are enabled via the ISER1 register (Section 6.6.2). Disabling interrupts is done through the ICER0 and ICER1 registers (Section 6.6.3 and Section 6.6.4).

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

Write — Writing 0 has no effect, writing 1 enables the interrupt.

Read — 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.

Table 68. Interrupt Set-Enable Register 0 register (ISER0 - address 0xE000 E100) bit description

Bit	Symbol	Description
0	ISE_PIO0_0	PIO0_0 start logic input interrupt enable.
1	ISE_PIO0_1	PIO0_1 start logic input interrupt enable.
2	ISE_PIO0_2	PIO0_2 start logic input interrupt enable.
3	ISE_PIO0_3	PIO0_3 start logic input interrupt enable.
4	ISE_PIO0_4	PIO0_4 start logic input interrupt enable.
5	ISE_PIO0_5	PIO0_5 start logic input interrupt enable.
6	ISE_PIO0_6	PIO0_6 start logic input interrupt enable.
7	ISE_PIO0_7	PIO0_7 start logic input interrupt enable.
8	ISE_PIO0_8	PIO0_8 start logic input interrupt enable.
9	ISE_PIO0_9	PIO0_9 start logic input interrupt enable.
10	ISE_PIO0_10	PIO0_10 start logic input interrupt enable.
11	ISE_PIO0_11	PIO0_11 start logic input interrupt enable.
12	ISE_PIO1_0	PIO1_0 start logic input interrupt enable.
13	ISE_PIO1_1	PIO1_1 start logic input interrupt enable.
14	ISE_PIO1_2	PIO1_2 start logic input interrupt enable.

Table 68. Interrupt Set-Enable Register 0 register (ISER0 - address 0xE000 E100) bit description *…continued*

Bit	Symbol	Description
15	ISE_PIO1_3	PIO1_3 start logic input interrupt enable.
16	ISE_PIO1_4	PIO1_4 start logic input interrupt enable.
17	ISE_PIO1_5	PIO1_5 start logic input interrupt enable.
18	ISE_PIO1_6	PIO1_6 start logic input interrupt enable.
19	ISE_PIO1_7	PIO1_7 start logic input interrupt enable.
20	ISE_PIO1_8	PIO1_8 start logic input interrupt enable.
21	ISE_PIO1_9	PIO1_9 start logic input interrupt enable.
22	ISE_PIO1_10	PIO1_10 start logic input interrupt enable.
23	ISE_PIO1_11	PIO1_11 start logic input interrupt enable.
24	ISE_PIO2_0	PIO2_0 start logic input interrupt enable.
25	ISE_PIO2_1	PIO2_1 start logic input interrupt enable.
26	ISE_PIO2_2	PIO2_2 start logic input interrupt enable.
27	ISE_PIO2_3	PIO2_3 start logic input interrupt enable.
28	ISE_PIO2_4	PIO2_4 start logic input interrupt enable.
29	ISE_PIO2_5	PIO2_5 start logic input interrupt enable.
30	ISE_PIO2_6	PIO2_6 start logic input interrupt enable.
31	ISE_PIO2_7	PIO2_7 start logic input interrupt enable.

6.6.2 Interrupt Set-Enable Register 1

The ISER1 register allows enabling the second group of peripheral interrupts, or for reading the enabled state of those interrupts. Disabling interrupts is done through the ICER0 and ICER1 registers (Section 6.6.3 and Section 6.6.4).

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

Write — Writing 0 has no effect, writing 1 enables the interrupt.

Read — 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.

Table 69. Interrupt Set-Enable Register 1 register (ISER1 - address 0xE000 E104) bit description

Bit	Symbol	Description
0	ISE_PIO2_8	PIO0_0 start logic input interrupt enable.
1	ISE_PIO2_9	PIO2_9 start logic input interrupt enable.
2	ISE_PIO2_10	PIO2_10 start logic input interrupt enable.
3	ISE_PIO2_11	PIO2_11 start logic input interrupt enable.
4	ISE_PIO3_0	PIO3_0 start logic input interrupt enable.
5	ISE_PIO3_1	PIO3_0 start logic input interrupt enable.
6	ISE_PIO3_2	PIO3_0 start logic input interrupt enable.
7	ISE_PIO3_3	PIO3_0 start logic input interrupt enable.
8	ISE_I2C0	I^{2}C0 interrupt enable.
9	ISE_CT16B0	Timer CT16B0 interrupt enable.
10	ISE_CT16B1	Timer CT16B1 interrupt enable.

Table 69. Interrupt Set-Enable Register 1 register (ISER1 - address 0xE000 E104) bit description *…continued*

Bit	Symbol	Description
11	ISE_CT32B0	Timer CT32B0 interrupt enable.
12	ISE_CT32B1	Timer CT32B1 interrupt enable.
13	ISE_SSP0	SSP0 interrupt enable.
14	ISE_UART	UART interrupt enable.
15	ISE_USBIRQ	USB IRQ interrupt enable.
16	ISE_USBFRQ	USB FRQ interrupt enable.
17	ISE_ADC	ADC interrupt enable.
18	ISE_WDT	WDT interrupt enable.
19	ISE_BOD	BOD interrupt enable.
20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
21	ISE_PIO_3	GPIO port 3 interrupt enable.
22	ISE_PIO_2	GPIO port 2 interrupt enable.
23	ISE_PIO_1	GPIO port 1 interrupt enable.
24	ISE_PIO_0	GPIO port 0 interrupt enable.
25	ISE_SSP1	SSP1 interrupt enable.
31:26	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

6.6.3 Interrupt Clear-Enable Register 0

The ICER0 register allows disabling the first 32 peripheral interrupts, or for reading the enabled state of those interrupts. The remaining interrupts are disabled via the ICER1 register (Section 6.6.4). Enabling interrupts is done through the ISER0 and ISER1 registers (Section 6.6.1 and Section 6.6.2).

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

Write — Writing 0 has no effect, writing 1 disables the interrupt.

Read — 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.

Table 70. Interrupt Clear-Enable Register 0

Bit	Symbol	Description
0	ICE_PIO0_0	PIO0_0 start logic input interrupt disable.
1	ICE_PIO0_1	PIO0_1 start logic input interrupt disable.
2	ICE_PIO0_2	PIO0_2 start logic input interrupt disable.
3	ICE_PIO0_3	PIO0_3 start logic input interrupt disable.
4	ICE_PIO0_4	PIO0_4 start logic input interrupt disable.
5	ICE_PIO0_5	PIO0_5 start logic input interrupt disable.
6	ICE_PIO0_6	PIO0_6 start logic input interrupt disable.
7	ICE_PIO0_7	PIO0_7 start logic input interrupt disable.
8	ICE_PIO0_8	PIO0_8 start logic input interrupt disable.
9	ICE_PIO0_9	PIO0_9 start logic input interrupt disable.
10	ICE_PIO0_10	PIO0_10 start logic input interrupt disable.

Table 70. Interrupt Clear-Enable Register 0 *…continued*

Bit	Symbol	Description
11	ICE_PIO0_11	PIO0_11 start logic input interrupt disable.
12	ICE_PIO1_0	PIO1_0 start logic input interrupt disable.
13	ICE_PIO1_1	PIO1_1 start logic input interrupt disable.
14	ICE_PIO1_2	PIO1_2 start logic input interrupt disable.
15	ICE_PIO1_3	PIO1_3 start logic input interrupt disable.
16	ICE_PIO1_4	PIO1_4 start logic input interrupt disable.
17	ICE_PIO1_5	PIO1_5 start logic input interrupt disable.
18	ICE_PIO1_6	PIO1_6 start logic input interrupt disable.
19	ICE_PIO1_7	PIO1_7 start logic input interrupt disable.
20	ICE_PIO1_8	PIO1_8 start logic input interrupt disable.
21	ICE_PIO1_9	PIO1_9 start logic input interrupt disable.
22	ICE_PIO1_10	PIO1_10 start logic input interrupt disable.
23	ICE_PIO1_11	PIO1_11 start logic input interrupt disable.
24	ICE_PIO2_0	PIO2_0 start logic input interrupt disable.
25	ICE_PIO2_1	PIO2_1 start logic input interrupt disable.
26	ICE_PIO2_2	PIO2_2 start logic input interrupt disable.
27	ICE_PIO2_3	PIO2_3 start logic input interrupt disable.
28	ICE_PIO2_4	PIO2_4 start logic input interrupt disable.
29	ICE_PIO2_5	PIO2_5 start logic input interrupt disable.
30	ICE_PIO2_6	PIO2_6 start logic input interrupt disable.
31	ICE_PIO2_7	PIO2_7 start logic input interrupt disable.

6.6.4 Interrupt Clear-Enable Register 1 register

The ICER1 register allows disabling the second group of peripheral interrupts, or for reading the enabled state of those interrupts. Enabling interrupts is done through the ISER0 and ISER1 registers (Section 6.6.1 and Section 6.6.2).

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

Write — Writing 0 has no effect, writing 1 disables the interrupt.

Read — 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.

Table 71. Interrupt Clear-Enable Register 1 register (ICER1 - address 0xE000 E184) bit description

Bit	Symbol	Description
0	ICE_PIO2_8	PIO0_0 start logic input interrupt disable.
1	ICE_PIO2_9	PIO2_9 start logic input interrupt disable.
2	ICE_PIO2_10	PIO2_10 start logic input interrupt disable.
3	ICE_PIO2_11	PIO2_11 start logic input interrupt disable.
4	ICE_PIO3_0	PIO3_0 start logic input interrupt disable.
5	ICE_PIO3_1	PIO3_0 start logic input interrupt disable.
6	ICE_PIO3_2	PIO3_0 start logic input interrupt disable.
7	ICE_PIO3_3	PIO3_0 start logic input interrupt disable.

Table 71. Interrupt Clear-Enable Register 1 register (ICER1 - address 0xE000 E184) bit description *…continued*

Bit	Symbol	Description
8	ICE_I2C0	I^{2}C0 interrupt disable.
9	ICE_CT16B0	Timer CT16B0 interrupt disable.
10	ICE_CT16B1	Timer CT16B1 interrupt disable.
11	ICE_CT32B0	Timer CT32B0 interrupt disable.
12	ICE_CT32B1	Timer CT32B1 interrupt disable.
13	ICE_SSP0	SSP0 interrupt disable.
14	ICE_UART	UART interrupt disable.
15	ICE_USBIRQ	USB IRQ interrupt disable.
16	ICE_USBFRQ	USB FRQ interrupt disable.
17	ICE_ADC	ADC interrupt disable.
18	ICE_WDT	WDT interrupt disable.
19	ICE_BOD	BOD interrupt disable.
20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
21	ICE_PIO_3	GPIO port 3 interrupt disable.
22	ICE_PIO_2	GPIO port 2 interrupt disable.
23	ICE_PIO_1	GPIO port 1 interrupt disable.
24	ICE_PIO_0	GPIO port 0 interrupt disable.
25	ICE_SSP1	SSP1 interrupt disable.
31:26	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

6.6.5 Interrupt Set-Pending Register 0 register

The ISPR0 register allows setting the pending state of the first 32 peripheral interrupts, or for reading the pending state of those interrupts. The remaining interrupts can have their pending state set via the ISPR1 register (Section 6.6.6). Clearing the pending state of interrupts is done through the ICPR0 and ICPR1 registers (Section 6.6.7 and Section 6.6.8).

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

Write — Writing 0 has no effect, writing 1 changes the interrupt state to pending.

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

Table 72. Interrupt Set-Pending Register 0 register (ISPR0 - address 0xE000 E200) bit description

Bit	Symbol	Description
0	ISP_PIO0_0	PIO0_0 start logic input interrupt pending set.
1	ISP_PIO0_1	PIO0_1 start logic input interrupt pending set.
2	ISP_PIO0_2	PIO0_2 start logic input interrupt pending set.
3	ISP_PIO0_3	PIO0_3 start logic input interrupt pending set.
4	ISP_PIO0_4	PIO0_4 start logic input interrupt pending set.

Table 72. Interrupt Set-Pending Register 0 register (ISPR0 - address 0xE000 E200) bit description *…continued*

Bit	Symbol	Description
5	ISP_PIO0_5	PIO0_5 start logic input interrupt pending set.
6	ISP_PIO0_6	PIO0_6 start logic input interrupt pending set.
7	ISP_PIO0_7	PIO0_7 start logic input interrupt pending set.
8	ISP_PIO0_8	PIO0_8 start logic input interrupt pending set.
9	ISP_PIO0_9	PIO0_9 start logic input interrupt pending set.
10	ISP_PIO0_10	PIO0_10 start logic input interrupt pending set.
11	ISP_PIO0_11	PIO0_11 start logic input interrupt pending set.
12	ISP_PIO1_0	PIO1_0 start logic input interrupt pending set.
13	ISP_PIO1_1	PIO1_1 start logic input interrupt pending set.
14	ISP_PIO1_2	PIO1_2 start logic input interrupt pending set.
15	ISP_PIO1_3	PIO1_3 start logic input interrupt pending set.
16	ISP_PIO1_4	PIO1_4 start logic input interrupt pending set.
17	ISP_PIO1_5	PIO1_5 start logic input interrupt pending set.
18	ISP_PIO1_6	PIO1_6 start logic input interrupt pending set.
19	ISP_PIO1_7	PIO1_7 start logic input interrupt pending set.
20	ISP_PIO1_8	PIO1_8 start logic input interrupt pending set.
21	ISP_PIO1_9	PIO1_9 start logic input interrupt pending set.
22	ISP_PIO1_10	PIO1_10 start logic input interrupt pending set.
23	ISP_PIO1_11	PIO1_11 start logic input interrupt pending set.
24	ISP_PIO2_0	PIO2_0 start logic input interrupt pending set.
25	ISP_PIO2_1	PIO2_1 start logic input interrupt pending set.
26	ISP_PIO2_2	PIO2_2 start logic input interrupt pending set.
27	ISP_PIO2_3	PIO2_3 start logic input interrupt pending set.
28	ISP_PIO2_4	PIO2_4 start logic input interrupt pending set.
29	ISP_PIO2_5	PIO2_5 start logic input interrupt pending set.
30	ISP_PIO2_6	PIO2_6 start logic input interrupt pending set.
31	ISP_PIO2_7	PIO2_7 start logic input interrupt pending set.

6.6.6 Interrupt Set-Pending Register 1 register

The ISPR1 register allows setting the pending state of the second group of peripheral interrupts, or for reading the pending state of those interrupts. Clearing the pending state of interrupts is done through the ICPR0 and ICPR1 registers (Section 6.6.7 and Section 6.6.8).

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

Write — Writing 0 has no effect, writing 1 changes the interrupt state to pending.

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

Table 73. Interrupt Set-Pending Register 1 register (ISPR1 - address 0xE000 E204) bit description

Bit	Symbol	Description
0	ISP_PIO2_8	PIO0_0 start logic input interrupt pending set.
1	ISP_PIO2_9	PIO2_9 start logic input interrupt pending set.
2	ISP_PIO2_10	PIO2_10 start logic input interrupt pending set.
3	ISP_PIO2_11	PIO2_11 start logic input interrupt pending set.
4	ISP_PIO3_0	PIO3_0 start logic input interrupt pending set.
5	ISP_PIO3_1	PIO3_0 start logic input interrupt pending set.
6	ISP_PIO3_2	PIO3_0 start logic input interrupt pending set.
7	ISP_PIO3_3	PIO3_0 start logic input interrupt pending set.
8	ISP_I2C0	I^{2}C0 interrupt pending set.
9	ISP_CT16B0	Timer CT16B0 interrupt pending set.
10	ISP_CT16B1	Timer CT16B1 interrupt pending set.
11	ISP_CT32B0	Timer CT32B0 interrupt pending set.
12	ISP_CT32B1	Timer CT32B1 interrupt pending set.
13	ISP_SSP0	SSP0 interrupt pending set.
14	ISP_UART	UART interrupt pending set.
15	ISP_USBIRQ	USB IRQ interrupt pending set.
16	ISP_USBFRQ	USB FRQ interrupt pending set.
17	ISP_ADC	ADC interrupt pending set.
18	ISP_WDT	WDT interrupt pending set.
19	ISP_BOD	BOD interrupt pending set.
20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
21	ISP_PIO_3	GPIO port 3 interrupt pending set.
22	ISP_PIO_2	GPIO port 2 interrupt pending set.
23	ISP_PIO_1	GPIO port 1 interrupt pending set.
24	ISP_PIO_0	GPIO port 0 interrupt pending set.
25	ISP_SSP1	SSP1 interrupt pending set.
31:26	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

6.6.7 Interrupt Clear-Pending Register 0 register

The ICPR0 register allows clearing the pending state of the first 32 peripheral interrupts, or for reading the pending state of those interrupts. The remaining interrupts can have their pending state cleared via the ICPR1 register (Section 6.6.8). Setting the pending state of interrupts is done through the ISPR0 and ISPR1 registers (Section 6.6.5 and Section 6.6.6).

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

Write — Writing 0 has no effect, writing 1 changes the interrupt state to not pending.

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

Table 74. Interrupt Clear-Pending Register 0 register (ICPR0 - address 0xE000 E280) bit description

Bit	Symbol	Function
0	ICP_PIO0_0	PIO0_0 start logic input interrupt pending clear.
1	ICP_PIO0_1	PIO0_1 start logic input interrupt pending clear.
2	ICP_PIO0_2	PIO0_2 start logic input interrupt pending clear.
3	ICP_PIO0_3	PIO0_3 start logic input interrupt pending clear.
4	ICP_PIO0_4	PIO0_4 start logic input interrupt pending clear.
5	ICP_PIO0_5	PIO0_5 start logic input interrupt pending clear.
6	ICP_PIO0_6	PIO0_6 start logic input interrupt pending clear.
7	ICP_PIO0_7	PIO0_7 start logic input interrupt pending clear.
8	ICP_PIO0_8	PIO0_8 start logic input interrupt pending clear.
9	ICP_PIO0_9	PIO0_9 start logic input interrupt pending clear.
10	ICP_PIO0_10	PIO0_10 start logic input interrupt pending clear.
11	ICP_PIO0_11	PIO0_11 start logic input interrupt pending clear.
12	ICP_PIO1_0	PIO1_0 start logic input interrupt pending clear.
13	ICP_PIO1_1	PIO1_1 start logic input interrupt pending clear.
14	ICP_PIO1_2	PIO1_2 start logic input interrupt pending clear.
15	ICP_PIO1_3	PIO1_3 start logic input interrupt pending clear.
16	ICP_PIO1_4	PIO1_4 start logic input interrupt pending clear.
17	ICP_PIO1_5	PIO1_5 start logic input interrupt pending clear.
18	ICP_PIO1_6	PIO1_6 start logic input interrupt pending clear.
19	ICP_PIO1_7	PIO1_7 start logic input interrupt pending clear.
20	ICP_PIO1_8	PIO1_8 start logic input interrupt pending clear.
21	ICP_PIO1_9	PIO1_9 start logic input interrupt pending clear.
22	ICP_PIO1_10	PIO1_10 start logic input interrupt pending clear.
23	ICP_PIO1_11	PIO1_11 start logic input interrupt pending clear.
24	ICP_PIO2_0	PIO2_0 start logic input interrupt pending clear.
25	ICP_PIO2_1	PIO2_1 start logic input interrupt pending clear.
26	ICP_PIO2_2	PIO2_2 start logic input interrupt pending clear.
27	ICP_PIO2_3	PIO2_3 start logic input interrupt pending clear.
28	ICP_PIO2_4	PIO2_4 start logic input interrupt pending clear.
29	ICP_PIO2_5	PIO2_5 start logic input interrupt pending clear.
30	ICP_PIO2_6	PIO2_6 start logic input interrupt pending clear.
31	ICP_PIO2_7	PIO2_7 start logic input interrupt pending clear.

6.6.8 Interrupt Clear-Pending Register 1 register

The ICPR1 register allows clearing the pending state of the second group of peripheral interrupts, or for reading the pending state of those interrupts. Setting the pending state of interrupts is done through the ISPR0 and ISPR1 registers (Section 6.6.5 and Section 6.6.6).

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

Write — Writing 0 has no effect, writing 1 changes the interrupt state to not pending.

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

Table 75. Interrupt Set-Pending Register 1 register (ISPR1 - address 0xE000 E204) bit description

Bit	Symbol	Description
0	ICP_PIO2_8	PIO0_0 start logic input interrupt pending clear.
1	ICP_PIO2_9	PIO2_9 start logic input interrupt pending clear.
2	ICP_PIO2_10	PIO2_10 start logic input interrupt pending clear.
3	ICP_PIO2_11	PIO2_11 start logic input interrupt pending clear.
4	ICP_PIO3_0	PIO3_0 start logic input interrupt pending clear.
5	ICP_PIO3_1	PIO3_0 start logic input interrupt pending clear.
6	ICP_PIO3_2	PIO3_0 start logic input interrupt pending clear.
7	ICP_PIO3_3	PIO3_0 start logic input interrupt pending clear.
8	ICP_I2C0	I^2C0 interrupt pending clear.
9	ICP_CT16B0	Timer CT16B0 interrupt pending clear.
10	ICP_CT16B1	Timer CT16B1 interrupt pending clear.
11	ICP_CT32B0	Timer CT32B0 interrupt pending clear.
12	ICP_CT32B1	Timer CT32B1 interrupt pending clear.
13	ICP_SSP0	SSP0 interrupt pending clear.
14	ICP_UART	UART interrupt pending clear.
15	ICP_USBIRQ	USB IRQ interrupt pending clear.
16	ICP_USBFRQ	USB FRQ interrupt pending clear.
17	ICP_ADC	ADC interrupt pending clear.
18	ICP_WDT	WDT interrupt pending clear.
19	ICP_BOD	BOD interrupt pending clear.
20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
21	ICP_PIO_3	GPIO port 3 interrupt pending clear.
22	ICP_PIO_2	GPIO port 2 interrupt pending clear.
23	ICP_PIO_1	GPIO port 1 interrupt pending clear.
24	ICP_PIO_0	GPIO port 0 interrupt pending clear.
25	ICP_SSP1	SSP1 interrupt pending clear.
31:26	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

6.6.9 Interrupt Active Bit Register 0

The IABR0 register is a read-only register that allows reading the active state of the first 32 peripheral interrupts. This allows determining which peripherals are asserting an interrupt to the NVIC, and may also be pending if there are enabled. The remaining interrupts can have their active state read via the IABR1 register (Section 6.6.10).

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

Write — n/a.

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

Table 76. Interrupt Active Bit Register 0 (IABR0 - address 0xE000 E300) bit description

Bit	Symbol	Function
0	IAB_PIO0_0	PIO0_0 start logic input interrupt active.
1	IAB_PIO0_1	PIO0_1 start logic input interrupt active.
2	IAB_PIO0_2	PIO0_2 start logic input interrupt active.
3	IAB_PIO0_3	PIO0_3 start logic input interrupt active.
4	IAB_PIO0_4	PIO0_4 start logic input interrupt active.
5	IAB_PIO0_5	PIO0_5 start logic input interrupt active.
6	IAB_PIO0_6	PIO0_6 start logic input interrupt active.
7	IAB_PIO0_7	PIO0_7 start logic input interrupt active.
8	IAB_PIO0_8	PIO0_8 start logic input interrupt active.
9	IAB_PIO0_9	PIO0_9 start logic input interrupt active.
10	IAB_PIO0_10	PIO0_10 start logic input interrupt active.
11	IAB_PIO0_11	PIO0_11 start logic input interrupt active.
12	IAB_PIO1_0	PIO1_0 start logic input interrupt active.
13	IAB_PIO1_1	PIO1_1 start logic input interrupt active.
14	IAB_PIO1_2	PIO1_2 start logic input interrupt active.
15	IAB_PIO1_3	PIO1_3 start logic input interrupt active.
16	IAB_PIO1_4	PIO1_4 start logic input interrupt active.
17	IAB_PIO1_5	PIO1_5 start logic input interrupt active.
18	IAB_PIO1_6	PIO1_6 start logic input interrupt active.
19	IAB_PIO1_7	PIO1_7 start logic input interrupt active.
20	IAB_PIO1_8	PIO1_8 start logic input interrupt active.
21	IAB_PIO1_9	PIO1_9 start logic input interrupt active.
22	IAB_PIO1_10	PIO1_10 start logic input interrupt active.
23	IAB_PIO1_11	PIO1_11 start logic input interrupt active.
24	IAB_PIO2_0	PIO2_0 start logic input interrupt active.
25	IAB_PIO2_1	PIO2_1 start logic input interrupt active.
26	IAB_PIO2_2	PIO2_2 start logic input interrupt active.
27	IAB_PIO2_3	PIO2_3 start logic input interrupt active.
28	IAB_PIO2_4	PIO2_4 start logic input interrupt active.
29	IAB_PIO2_5	PIO2_5 start logic input interrupt active.
30	IAB_PIO2_6	PIO2_6 start logic input interrupt active.
31	IAB_PIO2_7	PIO2_7 start logic input interrupt active.

6.6.10 Interrupt Active Bit Register 1

The IABR1 register is a read-only register that allows reading the active state of the second group of peripheral interrupts. This allows determining which peripherals are asserting an interrupt to the NVIC, and may also be pending if there are enabled.

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

Write — n/a.

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

Table 77. Interrupt Active Bit Register 1 (IABR1 - address 0xE000 E304) bit description

Bit	Symbol	Description
0	IAB_PIO2_8	PIO0_0 start logic input interrupt active.
1	IAB_PIO2_9	PIO2_9 start logic input interrupt active.
2	IAB_PIO2_10	PIO2_10 start logic input interrupt active.
3	IAB_PIO2_11	PIO2_11 start logic input interrupt active.
4	IAB_PIO3_0	PIO3_0 start logic input interrupt active.
5	IAB_PIO3_1	PIO3_0 start logic input interrupt active.
6	IAB_PIO3_2	PIO3_0 start logic input interrupt active.
7	IAB_PIO3_3	PIO3_0 start logic input interrupt active.
8	IAB_I2C0	I^{2}C0 interrupt active.
9	IAB_CT16B0	Timer CT16B0 interrupt active.
10	IAB_CT16B1	Timer CT16B1 interrupt active.
11	IAB_CT32B0	Timer CT32B0 interrupt active.
12	IAB_CT32B1	Timer CT32B1 interrupt active.
13	IAB_SSP0	SSP0 interrupt active.
14	IAB_UART	UART interrupt active.
15	IAB_USBIRQ	USB IRQ interrupt active.
16	IAB_USBFRQ	USB FRQ interrupt active.
17	IAB_ADC	ADC interrupt active.
18	IAB_WDT	WDT interrupt active.
19	IAB_BOD	BOD interrupt active.
20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
21	IAB_PIO_3	GPIO port 3 interrupt active.
22	IAB_PIO_2	GPIO port 2 interrupt active.
23	IAB_PIO_1	GPIO port 1 interrupt active.
24	IAB_PIO_0	GPIO port 0 interrupt active.
25	IAB_SSP1	SSP1 interrupt active.
31:26	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

6.6.11 Interrupt Priority Register 0

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

Table 78. Interrupt Priority Register 0 (IPR0 - address 0xE000 E400) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	IP_PIO0_0	PIO0_0 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_PIO0_1	PIO0_1 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
20:16	Unimplemented	These bits ignore writes, and read as 0.
23:21	IP_PIO0_2	PIO0_2 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
28:24	Unimplemented	These bits ignore writes, and read as 0.
31:29	IP_PIO0_3	PIO0_3 Interrupt Priority. 0 = highest priority. 7 = lowest priority.

6.6.12 Interrupt Priority Register 1

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

Table 79. Interrupt Priority Register 1 (IPR1 - address 0xE000 E404) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	IP_PIO0_4	PIO0_4 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_PIO0_5	PIO0_5 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
20:16	Unimplemented	These bits ignore writes, and read as 0.
23:21	IP_PIO0_6	PIO0_6 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
28:24	Unimplemented	These bits ignore writes, and read as 0.
31:29	IP_PIO0_7	PIO0_7 Interrupt Priority. 0 = highest priority. 7 = lowest priority.

All information provided in this document is subject to legal disclaimers.

6.6.13 Interrupt Priority Register 2

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

Table 80. Interrupt Priority Register 2 (IPR2 - address 0xE000 E408) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	IP_PIO0_8	PIO0_8 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_PIO0_9	PIO0_9 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
20:16	Unimplemented	These bits ignore writes, and read as 0.
23:21	IP_PIO0_10	PIO0_10 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
28:24	Unimplemented	These bits ignore writes, and read as 0.
31:29	IP_PIO0_11	PIO0_11 Interrupt Priority. 0 = highest priority. 7 = lowest priority.

6.6.14 Interrupt Priority Register 3

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

Table 81. Interrupt Priority Register 3 (IPR3 - address 0xE000 E40C) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	IP_PIO1_0	PIO1_0 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_PIO1_1	PIO1_1 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
20:16	Unimplemented	These bits ignore writes, and read as 0.
23:21	IP_PIO1_2	PIO1_2 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
28:24	Unimplemented	These bits ignore writes, and read as 0.
31:29	IP_PIO1_3	PIO1_3 Interrupt Priority. 0 = highest priority. 7 = lowest priority.

6.6.15 Interrupt Priority Register 4

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

Table 82. Interrupt Priority Register 4 (IPR4 - address 0xE000 E410) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	IP_PIO1_4	PIO0_4 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_PIO1_5	PIO0_5 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
20:16	Unimplemented	These bits ignore writes, and read as 0.
23:21	IP_PIO1_6	PIO0_6 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
28:24	Unimplemented	These bits ignore writes, and read as 0.
31:29	IP_PIO1_7	PIO0_7 Interrupt Priority. 0 = highest priority. 7 = lowest priority.

6.6.16 Interrupt Priority Register 5

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

Table 83. Interrupt Priority Register 5 (IPR5 - address 0xE000 E414) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	IP_PIO1_8	PIO1_8 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_PIO1_9	PIO1_9 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
20:16	Unimplemented	These bits ignore writes, and read as 0.
23:21	IP_PIO1_10	PIO1_10 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
28:24	Unimplemented	These bits ignore writes, and read as 0.
31:29	IP_PIO1_11	PIO1_11 Interrupt Priority. 0 = highest priority. 7 = lowest priority.

6.6.17 Interrupt Priority Register 6

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

Table 84. Interrupt Priority Register 60 (IPR6 - address 0xE000 E418) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	IP_PIO2_0	PIO2_0 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_PIO2_1	PIO2_1 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
20:16	Unimplemented	These bits ignore writes, and read as 0.
23:21	IP_PIO2_2	PIO2_2 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
28:24	Unimplemented	These bits ignore writes, and read as 0.
31:29	IP_PIO2_3	PIO2_3 Interrupt Priority. 0 = highest priority. 7 = lowest priority.

6.6.18 Interrupt Priority Register 7

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

Table 85. Interrupt Priority Register 7 (IPR7 - address 0xE000 E41C) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	IP_PIO2_4	PIO2_4 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_PIO2_5	PIO2_5 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
20:16	Unimplemented	These bits ignore writes, and read as 0.
23:21	IP_PIO2_6	PIO2_6 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
28:24	Unimplemented	These bits ignore writes, and read as 0.
31:29	IP_PIO2_7	PIO2_7 Interrupt Priority. 0 = highest priority. 7 = lowest priority.

6.6.19 Interrupt Priority Register 8

The IPR8 register controls the priority of four peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 86. Interrupt Priority Register 8 (IPR8 - address 0xE000 E420) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	IP_PIO2_8	PIO0_8 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_PIO2_9	PIO0_9 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
20:16	Unimplemented	These bits ignore writes, and read as 0.
23:21	IP_PIO2_10	PIO0_10 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
28:24	Unimplemented	These bits ignore writes, and read as 0.
31:29	IP_PIO2_11	PIO0_11 Interrupt Priority. 0 = highest priority. 7 = lowest priority.

6.6.20 Interrupt Priority Register 9

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

Table 87. Interrupt Priority Register 9 (IPR9 - address 0xE000 E424) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	IP_PIO3_0	PIO3_0 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_PIO3_1	PIO3_1 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
20:16	Unimplemented	These bits ignore writes, and read as 0.
23:21	IP_PIO3_2	PIO3_2 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
28:24	Unimplemented	These bits ignore writes, and read as 0.
31:29	IP_PIO3_3	PIO3_3 Interrupt Priority. 0 = highest priority. 7 = lowest priority.

6.6.21 Interrupt Priority Register 10

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

Table 88. Interrupt Priority Register 10 (IPR10 - address 0xE000 E428) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	IP_I2C	I2C Interrupt Priority. 0 = highest priority. 7 = lowest priority.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_CT16B0	CT16B0 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
20:16	Unimplemented	These bits ignore writes, and read as 0.
23:21	IP_CT16B1	CT16B1 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
28:24	Unimplemented	These bits ignore writes, and read as 0.
31:29	IP_CT32B0	CT32B0 Interrupt Priority. 0 = highest priority. 7 = lowest priority.

6.6.22 Interrupt Priority Register 11

The IPR11 register controls the priority of four peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 89. Interrupt Priority Register 11 (IPR11 - address 0xE000 E42C) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	IP_CT32B1	CT32B1 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_SSP0	SSP0 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
20:16	Unimplemented	These bits ignore writes, and read as 0.
23:21	IP_UART	UART Interrupt Priority. 0 = highest priority. 7 = lowest priority.
28:24	Unimplemented	These bits ignore writes, and read as 0.
31:29	IP_USBIRQ	USBIRQ Interrupt Priority. 0 = highest priority. 7 = lowest priority.

6.6.23 Interrupt Priority Register 12

The IPR12 register controls the priority of four peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 90. Interrupt Priority Register 12 (IPR12 - address 0xE000 E430) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	IP_USNFIQ	USBFIQ Interrupt Priority. 0 = highest priority. 7 = lowest priority.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_ADC	ADC Interrupt Priority. 0 = highest priority. 7 = lowest priority.
20:16	Unimplemented	These bits ignore writes, and read as 0.
23:21	IP_WDT	WDT Interrupt Priority. 0 = highest priority. 7 = lowest priority.
28:24	Unimplemented	These bits ignore writes, and read as 0.
31:29	IP_BOD	BOD Interrupt Priority. 0 = highest priority. 7 = lowest priority.

6.6.24 Interrupt Priority Register 13

The IPR13 register controls the priority of four peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 91. Interrupt Priority Register 13 (IPR13 - address 0xE000 E434) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	-	Reserved.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_PIO3	PIO3 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
20:16	Unimplemented	These bits ignore writes, and read as 0.
23:21	IP_PIO2	PIO2 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
28:24	Unimplemented	These bits ignore writes, and read as 0.
31:29	IP_PIO1	PIO1 Interrupt Priority. 0 = highest priority. 7 = lowest priority.

6.6.25 Interrupt Priority Register 14

The IPR14 register controls the priority of four peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 92. Interrupt Priority Register 14 (IPR14 - address 0xE000 E438) bit description

Bit	Symbol	Description
4:0	Unimplemented	These bits ignore writes, and read as 0.
7:5	IP_PIO0	PIO0 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
12:8	Unimplemented	These bits ignore writes, and read as 0.
15:13	IP_SSP1	SSP1 Interrupt Priority. 0 = highest priority. 7 = lowest priority.
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

6.6.26 Software Trigger Interrupt Register

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

By default, only privileged software can write to the STIR register. Unprivileged software can be given this ability if privileged software sets the USERSETMPEND bit in the ARM Cortex-M3 CCR register.

Table 93. Software Trigger Interrupt Register (STIR - address 0xE000 EF00) bit description

Bit	Symbol	Description
8:0	INTID	Writing a value to this field generates an interrupt for the specified the interrupt number (see Table 66). The range allowed for the LPC13xx is 0 to 57.
31:9	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

UM10375

Chapter 7: LPC13xx I/O configuration

Rev. 5 — 21 June 2012 **User manual**

7.1 How to read this chapter

The implementation of the I/O configuration registers varies for different LPC13xx parts and packages. See Table 94 and Table 96 for IOCON registers and register bits which are not used in all parts or packages.

For the LPC1311/01 and LPC1313/01, a pseudo open-drain mode can be selected in the IOCON registers for each digital pins except the I2C pins. The open-drain mode is not available in the LPC1311/13/42/43 parts.

Table 94. Availability of IOCON registers

Part	IOCON_PIO2_1 to IOCON_PIO2_11	IOCON_PIO3_0, IOCON_PIO3_1, IOCON_PIO3_3	IOCON_PIO3_4, IOCON_PIO3_5	USB function [1]	SSP1 function [2]	UART additional modem function [3]	UART location registers [4]
LPC1311FHN33	no	no	yes	no	no	no	no
LPC1311FHN33/01	no	no	yes	no	no	no	no
LPC1313FHN33	no	no	yes	no	no	no	no
LPC1313FHN33/01	no	no	yes	no	no	no	no
LPC1313FBD48	yes	yes	yes	no	no	no	no
LPC1313FBD48/01	yes	yes	yes	no	yes	yes	yes
LPC1342FHN33	no	no	no	yes	no	no	no
LPC1342FBD48	yes	yes	no	yes	no	no	no
LPC1343FHN33	no	no	no	yes	no	no	no
LPC1343FBD48	yes	yes	no	yes	no	no	no

[1] In registers IOCON_PIO0_1, IOCON_PIO0_3, IOCON_PIO0_6

[2] In registers IOCON_PIO2_0, IOCON_PIO2_1, IOCON_PIO2_2, IOCON_PIO2_3

[3] In registers IOCON_PIO3_0, IOCON_PIO3_1, IOCON_PIO3_2, IOCON_PIO3_3

[4] IOCON_DSR, IOCON_DCD, IOCON_RI registers

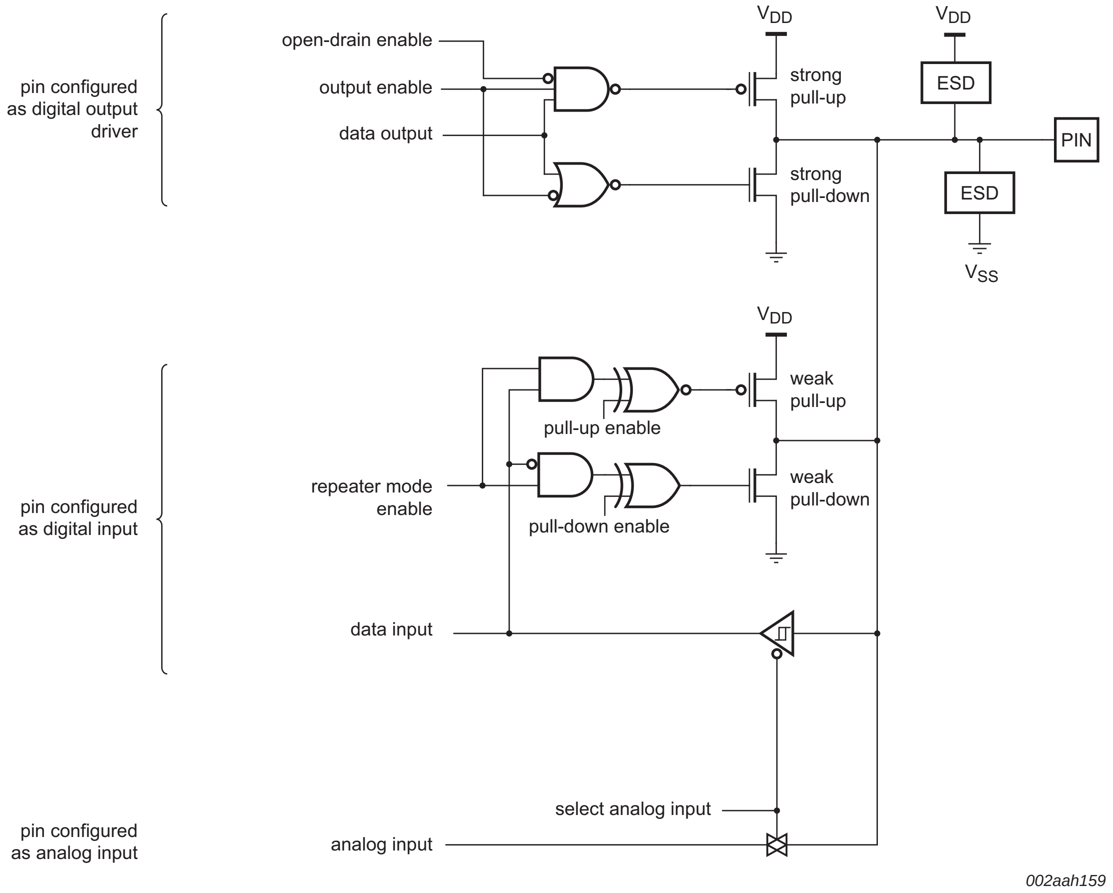
7.2 Introduction

The I/O configuration registers control the electrical characteristics of the pins. The following characteristics are configurable:

- pin function
- internal pull-up/pull-down or Repeater mode function
- hysteresis
- analog input or digital mode for pins hosting the ADC inputs
- I^{2}C mode for pins hosting the I^{2}C-bus function

7.3 General description

The IOCON registers control the function (GPIO or peripheral function), the input mode, and the hysteresis of all PIO pins. In addition, the I^{2}C-bus pins can be configured for different I^{2}C-bus modes. If a pin is used as input pin for the ADC, an analog input mode can be selected.



See Section 7.1 for open-drain mode.

Fig 9. Standard I/O pin configuration

7.3.1 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 GPIO pins, the GPIODIR registers determine whether the pin is configured as an input or output (see Table 150). For any peripheral function, the pin direction is controlled automatically depending on the pin's functionality. The GPIODIR registers have no effect on peripheral functions.

7.3.2 Pin mode

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

The possible on-chip resistor configurations are pull-up enabled, pull-down enabled, or no pull-up/pull-down. The default value is pull-up enabled. The pins are pulled up to 2.6 V for LPC1311/13/42/43 parts and pulled up to 3.3 V for LPC1311/01 and LPC1313/01 parts (V_{DD} = 3.3 V).

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

7.3.3 Hysteresis

The input buffer for digital functions can be configured with hysteresis or as plain buffer through the IOCON registers (see the *LPC1311/13/43/44 data sheet* for details).

If the external pad supply voltage V_{DD} is between 2.5 V and 3.6 V, the hysteresis buffer can be enabled or disabled. If V_{DD} is below 2.5 V, the hysteresis buffer must be **disabled** to use the pin in input mode.

7.3.4 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, Hysteresis and Pin mode settings have no effect.

7.3.5 Open-drain Mode

When output is selected, either by selecting a special function in the FUNC field, or by selecting GPIO function for a pin having a 1 in its GPIODIR register, a 1 in the OD bit selects open-drain operation, that is, a 1 disables the high-drive transistor. This option has no effect on the primary I^2C pins.

Remark: The open-drain mode is only available on parts LPC1311/01 and LPC1313/01.

7.3.6 I^2C mode

If the I^2C function is selected by the FUNC bits of registers IOCON_PIO0_4 (Table 107) and IOCON_PIO0_5 (Table 108), then the I^2C-bus pins can be configured for different I^2C-modes:

- Standard mode/Fast-mode I^2C with input glitch filter (this includes an open-drain output according to the I^2C-bus specification).
- Fast-mode Plus with input glitch filter (this includes an open-drain output according to the I^2C-bus specification). In this mode, the pins function as high-current sinks.
- Standard, open-drain I/O functionality without input filter.

Remark: Either Standard mode/Fast-mode I^2C or Standard I/O functionality should be selected if the pin is used as GPIO pin.

7.4 Register description

The I/O configuration registers control the following pins: PIO ports, the I^{2}C-bus pins, and the ADC input pins.

The pin functions selectable in each IOCON register are listed in order (function 0/function 1/function 2/...) in the description column in Table 95.

Remark: The IOCON registers are listed in order of their memory locations in Table 95 which correspond to the order of their physical pin numbers in the LQFP48 package starting at the upper left corner with pin 1 (PIO2_6). See Table 96 for a listing of IOCON registers ordered by port number.

Table 95. Register overview: I/O configuration block (base address 0x4004 4000)

Name	Access	Address offset	Description	Reset value
IOCON_PIO2_6	R/W	0x000	I/O configuration for pin PIO2_6	0xD0
-	R/W	0x004	Reserved	-
IOCON_PIO2_0	R/W	0x008	I/O configuration for pin PIO2_0/DTR/SSEL1	0xD0
IOCON_RESET_PIO0_0	R/W	0x00C	I/O configuration for pin RESET/PIO0_0	0xD0
IOCON_PIO0_1	R/W	0x010	I/O configuration for pin PIO0_1/CLKOUT/ CT32B0_MAT2/USB_FTOGGLE	0xD0
IOCON_PIO1_8	R/W	0x014	I/O configuration for pin PIO1_8/CT16B1_CAP0	0xD0
-	R/W	0x018	Reserved	-
IOCON_PIO0_2	R/W	0x01C	I/O configuration for pin PIO0_2/SSEL0/ CT16B0_CAP0	0xD0
IOCON_PIO2_7	R/W	0x020	I/O configuration for pin PIO2_7	0xD0
IOCON_PIO2_8	R/W	0x024	I/O configuration for pin PIO2_8	0xD0
IOCON_PIO2_1	R/W	0x028	I/O configuration for pin PIO2_1/DSR/SCK1	0xD0
IOCON_PIO0_3	R/W	0x02C	I/O configuration for pin PIO0_3/USB_VBUS	0xD0
IOCON_PIO0_4	R/W	0x030	I/O configuration for pin PIO0_4/SCL	0x00
IOCON_PIO0_5	R/W	0x034	I/O configuration for pin PIO0_5/SDA	0x00
IOCON_PIO1_9	R/W	0x038	I/O configuration for pin PIO1_9/CT16B1_MAT0	0xD0
IOCON_PIO3_4	R/W	0x03C	I/O configuration for pin PIO3_4	0xD0
IOCON_PIO2_4	R/W	0x040	I/O configuration for pin PIO2_4	0xD0
IOCON_PIO2_5	R/W	0x044	I/O configuration for pin PIO2_5	0xD0
IOCON_PIO3_5	R/W	0x048	I/O configuration for pin PIO3_5	0xD0
IOCON_PIO0_6	R/W	0x04C	I/O configuration for pin PIO0_6/USB_CONNECT/SCK	0xD0
IOCON_PIO0_7	R/W	0x050	I/O configuration for pin PIO0_7/CTS	0xD0
IOCON_PIO2_9	R/W	0x054	I/O configuration for pin PIO2_9	0xD0
IOCON_PIO2_10	R/W	0x058	I/O configuration for pin PIO2_10	0xD0
IOCON_PIO2_2	R/W	0x05C	I/O configuration for pin PIO2_2/DCD/MISO1	0xD0
IOCON_PIO0_8	R/W	0x060	I/O configuration for pin PIO0_8/MISO0/CT16B0_MAT0	0xD0
IOCON_PIO0_9	R/W	0x064	I/O configuration for pin PIO0_9/MOSI0/ CT16B0_MAT1/SWO	0xD0
IOCON_SWCLK_PIO0_10	R/W	0x068	I/O configuration for pin SWCLK/PIO0_10/ SCK/CT16B0_MAT2	0xD0

Table 95. Register overview: I/O configuration block (base address 0x4004 4000) *…continued*

Name	Access	Address offset	Description	Reset value
IOCON_PIO1_10	R/W	0x06C	I/O configuration for pin PIO1_10/AD6/ CT16B1_MAT1	0xD0
IOCON_PIO2_11	R/W	0x070	I/O configuration for pin PIO2_11/SCK	0xD0
IOCON_R_PIO0_11	R/W	0x074	I/O configuration for pin R/PIO0_11/AD0/CT32B0_MAT3	0xD0
IOCON_R_PIO1_0	R/W	0x078	I/O configuration for pin R/PIO1_0/AD1/ CT32B1_CAP0	0xD0
IOCON_R_PIO1_1	R/W	0x07C	I/O configuration for pin R/PIO1_1/AD2/CT32B1_MAT0	0xD0
IOCON_R_PIO1_2	R/W	0x080	I/O configuration for pin R/PIO1_2/AD3/ CT32B1_MAT1	0xD0
IOCON_PIO3_0	R/W	0x084	I/O configuration for pin PIO3_0/$\overline{\text{DTR}}$	0xD0
IOCON_PIO3_1	R/W	0x088	I/O configuration for pin PIO3_1/$\overline{\text{DSR}}$	0xD0
IOCON_PIO2_3	R/W	0x08C	I/O configuration for pin PIO2_3/$\overline{\text{RI}}$/MOSI1	0xD0
IOCON_SWDIO_PIO1_3	R/W	0x090	I/O configuration for pin SWDIO/PIO1_3/AD4/ CT32B1_MAT2	0xD0
IOCON_PIO1_4	R/W	0x094	I/O configuration for pin PIO1_4/AD5/CT32B1_MAT3	0xD0
IOCON_PIO1_11	R/W	0x098	I/O configuration for pin PIO1_11/AD7	0xD0
IOCON_PIO3_2	R/W	0x09C	I/O configuration for pin PIO3_2/$\overline{\text{DCD}}$	0xD0
IOCON_PIO1_5	R/W	0x0A0	I/O configuration for pin PIO1_5/$\overline{\text{RTS}}$/CT32B0_CAP0	0xD0
IOCON_PIO1_6	R/W	0x0A4	I/O configuration for pin PIO1_6/RXD/CT32B0_MAT0	0xD0
IOCON_PIO1_7	R/W	0x0A8	I/O configuration for pin PIO1_7/TXD/CT32B0_MAT1	0xD0
IOCON_PIO3_3	R/W	0x0AC	I/O configuration for pin PIO3_3/$\overline{\text{RI}}$	0xD0
IOCON_SCK0_LOC	R/W	0x0B0	SCK0 pin location register	0
IOCON_DSR_LOC	R/W	0x0B4	$\overline{\text{DSR}}$ pin location select register	0
IOCON_DCD_LOC	R/W	0x0B8	$\overline{\text{DCD}}$ pin location select register	0
IOCON_RI_LOC	R/W	0x0BC	$\overline{\text{RI}}$ pin location register	0

Table 96. I/O configuration registers ordered by port number

Port pin	Pin name	LQFP48	HVQFN33	Reference
PIO0_0	IOCON_RESET_PIO0_0	yes	yes	Table 99
PIO0_1	IOCON_PIO0_1	yes	yes	Table 100
PIO0_2	IOCON_PIO0_2	yes	yes	Table 102
PIO0_3	IOCON_PIO0_3	yes	yes	Table 106
PIO0_4	IOCON_PIO0_4	yes	yes	Table 107
PIO0_5	IOCON_PIO0_5	yes	yes	Table 108
PIO0_6	IOCON_PIO0_6	yes	yes	Table 114
PIO0_7	IOCON_PIO0_7	yes	yes	Table 115
PIO0_8	IOCON_PIO0_8	yes	yes	Table 119
PIO0_9	IOCON_PIO0_9	yes	yes	Table 120
PIO0_10	IOCON_SWCLK_PIO0_10	yes	yes	Table 121
PIO0_11	IOCON_R_PIO0_11	yes	yes	Table 124
PIO1_0	IOCON_R_PIO1_0	yes	yes	Table 125

Table 96. I/O configuration registers ordered by port number *…continued*

Port pin	Pin name	LQFP48	HVQFN33	Reference
PIO1_1	IOCON_R_PIO1_1	yes	yes	Table 126
PIO1_2	IOCON_R_PIO1_2	yes	yes	Table 127
PIO1_3	IOCON_SWDIO_PIO1_3	yes	yes	Table 131
PIO1_4	IOCON_PIO1_4	yes	yes	Table 132
PIO1_5	IOCON_PIO1_5	yes	yes	Table 135
PIO1_6	IOCON_PIO1_6	yes	yes	Table 136
PIO1_7	IOCON_PIO1_7	yes	yes	Table 137
PIO1_8	IOCON_PIO1_8	yes	yes	Table 101
PIO1_9	IOCON_PIO1_9	yes	yes	Table 109
PIO1_10	IOCON_PIO1_10	yes	yes	Table 122
PIO1_11	IOCON_PIO1_11	yes	yes	Table 133
PIO2_0	IOCON_PIO2_0	yes	yes	Table 98
PIO2_1	IOCON_PIO2_1	yes	no	Table 105
PIO2_2	IOCON_PIO2_2	yes	no	Table 118
PIO2_3	IOCON_PIO2_3	yes	no	Table 130
PIO2_4	IOCON_PIO2_4	yes	no	Table 111
PIO2_5	IOCON_PIO2_5	yes	no	Table 112
PIO2_6	IOCON_PIO2_6	yes	no	Table 97
PIO2_7	IOCON_PIO2_7	yes	no	Table 103
PIO2_8	IOCON_PIO2_8	yes	no	Table 104
PIO2_9	IOCON_PIO2_9	yes	no	Table 116
PIO2_10	IOCON_PIO2_10	yes	no	Table 117
PIO2_11	IOCON_PIO2_11	yes	no	Table 123
PIO3_0	IOCON_PIO3_0	yes	no	Table 128
PIO3_1	IOCON_PIO3_1	yes	no	Table 129
PIO3_2	IOCON_PIO3_2	yes	yes	Table 134
PIO3_3	IOCON_PIO3_3	yes	no	Table 138
PIO3_4	IOCON_PIO3_4	yes, on LPC1311/13 and LPC1311/13/01. [1]	yes, on LPC1311/13 and LPC1311/13/01. [1]	Table 110
PIO3_5	IOCON_PIO3_5	yes, on LPC1311/13[1]	yes, on LPC1311/13[1]	Table 113

[1] On LPC134x, PIO3_4 and PIO3_5 are not available. The corresponding pins are used for the USB D+ and D– functions.

7.4.1 IOCON_PIO2_6

Table 97. IOCON_PIO2_6 register (IOCON_PIO2_6, address 0x4004 4000) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO2_6.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.2 IOCON_PIO2_0

Table 98. IOCON_PIO2_0 register (IOCON_PIO2_0, address 0x4004 4008) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Select function PIO2_0.	
		0x1	Select function $\overline{\text{DTR}}$.	
		0x2	Select function SSEL1 (function not available on all parts).	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.3 IOCON_nRESET_PIO0_0

Table 99. IOCON_nRESET_PIO0_0 register (IOCON_nRESET_PIO0_0, address 0x4004 400C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function $\overline{\text{RESET}}$.	
		0x1	Selects function PIO0_0.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.4 IOCON_PIO0_1

Table 100. IOCON_PIO0_1 register (IOCON_PIO0_1, address 0x4004 4010) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO0_1.	
		0x1	Selects function CLKOUT.	
		0x2	Selects function CT32B0_MAT2.	
		0x3	Selects function USB_FTOGGLE (function not available on all parts)	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011

Table 100. IOCON_PIO0_1 register (IOCON_PIO0_1, address 0x4004 4010) bit description

Bit	Symbol	Value	Description	Reset value
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.5 IOCON_PIO1_8

Table 101. IOCON_PIO1_8 register (IOCON_PIO1_8, address 0x4004 4014) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO1_8.	
		0x1	Selects function CT16B1_CAP0.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.6 IOCON_PIO0_2

Table 102. IOCON_PIO0_2 register (IOCON_PIO0_2, address 0x4004 401C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO0_2.	
		0x1	Selects function SSEL0.	
		0x2	Selects function CT16B0_CAP0.	

Table 102. IOCON_PIO0_2 register (IOCON_PIO0_2, address 0x4004 401C) bit description

Bit	Symbol	Value	Description	Reset value
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.7 IOCON_PIO2_7

Table 103. IOCON_PIO2_7 register (IOCON_PIO2_7, address 0x4004 4020) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO2_7.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.8 IOCON_PIO2_8

Table 104. IOCON_PIO2_8 register (IOCON_PIO2_8, address 0x4004 4024) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO2_8.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.9 IOCON_PIO2_1

Table 105. IOCON_PIO2_1 register (IOCON_PIO2_1, address 0x4004 4028) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO2_1.	
		0x1	Select function $\overline{\mathrm{DSR}}$.	
		0x2	Select function SCK1 (function not available on all parts).	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.10 IOCON_PIO0_3

Table 106. IOCON_PIO0_3 register (IOCON_PIO0_3, address 0x4004 402C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO0_3.	
		0x1	Selects function USB_VBUS (function not available on all parts).	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.11 IOCON_PIO0_4

Table 107. IOCON_PIO0_4 register (IOCON_PIO0_4, address 0x4004 4030) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO0_4 (open-drain pin).	
		0x1	Selects I2C function SCL (open-drain pin).	
7:3	-	-	Reserved	00000
9:8	I2CMODE		Selects I2C mode. Selects I2C mode. Select Standard mode (I2CMODE = 00, default) or Standard I/O functionality (I2CMODE = 01) if the pin function is GPIO (FUNC = 000).	00
		0x0	Standard mode/ Fast-mode I2C	
		0x1	Standard I/O functionality	
		0x2	Fast-mode Plus I2C	
		0x3	Reserved	
31:10	-	-	Reserved	-

7.4.12 IOCON_PIO0_5

Table 108. IOCON_PIO0_5 register (IOCON_PIO0_5, address 0x4004 4034) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO0_5 (open-drain pin).	
		0x1	Selects I2C function SDA (open-drain pin).	
7:3	-	-	Reserved	00000
9:8	I2CMODE		Selects I2C mode. Select Standard mode (I2CMODE = 00, default) or Standard I/O functionality (I2CMODE = 01) if the pin function is GPIO (FUNC = 000).	00
		0x0	Standard mode/ Fast-mode I2C	
		0x1	Standard I/O functionality	
		0x2	Fast-mode Plus I2C	
		0x3	Reserved	
31:10	-	-	Reserved	-

7.4.13 IOCON_PIO1_9

Table 109. IOCON_PIO1_9 register (IOCON_PIO1_9, address 0x4004 4038) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO1_9.	
		0x1	Selects function CT16B1_MAT0.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.14 IOCON_PIO3_4

Table 110. IOCON_PIO3_4 register (IOCON_PIO3_4, address 0x4004 403C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO3_4.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.15 IOCON_PIO2_4

Table 111. IOCON_PIO2_4 register (IOCON_PIO2_4, address 0x4004 4040) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO2_4.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.16 IOCON_PIO2_5

Table 112. IOCON_PIO2_5 register (IOCON_PIO2_5, address 0x4004 4044) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO2_5.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.17 IOCON_PIO3_5

Table 113. IOCON_PIO3_5 register (IOCON_PIO3_5, address 0x4004 4048) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO3_5.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.18 IOCON_PIO0_6

Table 114. IOCON_PIO0_6 register (IOCON_PIO0_6, address 0x4004 404C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO0_6.	
		0x1	Selects function USB_CONNECT (function not available on all parts).	
		0x2	Selects function SCK0 (only if pin PIO0_6/USB_CONNECT/SCK0 selected in Table 139).	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.19 IOCON_PIO0_7

Table 115. IOCON_PIO0_7 register (IOCON_PIO0_7, address 0x4004 4050) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO0_7.	
		0x1	Select function CTS.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011

Table 115. IOCON_PIO0_7 register (IOCON_PIO0_7, address 0x4004 4050) bit description

Bit	Symbol	Value	Description	Reset value
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.20 IOCON_PIO2_9

Table 116. IOCON_PIO2_9 register (IOCON_PIO2_9, address 0x4004 4054) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO2_9.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.21 IOCON_PIO2_10

Table 117. IOCON_PIO2_10 register (IOCON_PIO2_10, address 0x4004 4058) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO2_10.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	

Table 117. IOCON_PIO2_10 register (IOCON_PIO2_10, address 0x4004 4058) bit description

Bit	Symbol	Value	Description	Reset value
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.22 IOCON_PIO2_2

Table 118. IOCON_PIO2_2 register (IOCON_PIO2_2, address 0x4004 405C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO2_2.	
		0x1	Select function $\overline{\text{DCD}}$.	
		0x2	Select function MISO1 (function not available on all parts).	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.23 IOCON_PIO0_8

Table 119. IOCON_PIO0_8 register (IOCON_PIO0_8, address 0x4004 4060) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO0_8.	
		0x1	Selects function MISO0.	
		0x2	Selects function CT16B0_MAT0.	

Table 119. IOCON_PIO0_8 register (IOCON_PIO0_8, address 0x4004 4060) bit description

Bit	Symbol	Value	Description	Reset value
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.24 IOCON_PIO0_9

Table 120. IOCON_PIO0_9 register (IOCON_PIO0_9, address 0x4004 4064) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO0_9.	
		0x1	Selects function MOSI0.	
		0x2	Selects function CT16B0_MAT1.	
		0x3	Selects function SWO	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.25 IOCON_SWCLK_PIOO_10

Table 121. IOCON_SWCLK_PIO0_10 register (IOCON_SWCLK_PIO0_10, address 0x4004 4068) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function SWCLK.	
		0x1	Selects function PIO0_10.	
		0x2	Selects function SCK0 (only if pin SWCLK/PIO0_10/SCK0/CT16B0_MAT2 selected in Table 139).	
		0x3	Selects function CT16B0_MAT2.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.26 IOCON_PIO1_10

Table 122. IOCON_PIO1_10 register (IOCON_PIO1_10, address 0x4004 406C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO1_10.	
		0x1	Selects function AD6.	
		0x2	Selects function CT16B1_MAT1.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	

Table 122. IOCON_PIO1_10 register (IOCON_PIO1_10, address 0x4004 406C) bit description

Bit	Symbol	Value	Description	Reset value
6	-	-	Reserved	1
7	ADMODE		Selects Analog/Digital mode	1
		0	Analog input mode	
		1	Digital functional mode	
9:8	-	-	Reserved	00
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.27 IOCON_PIO2_11

Table 123. IOCON_PIO2_11 register (IOCON_PIO2_11, address 0x4004 4070) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO2_11.	
		0x1	Selects function SCK0 (only if pin PIO2_11/SCK0 selected in Table 139)	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.28 IOCON_R_PIO0_11

Table 124. IOCON_R_PIO0_11 register (IOCON_R_PIO0_11, address 0x4004 4074) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function R. This function is reserved. Select one of the alternate functions below.	
		0x1	Selects function PIO0_11.	
		0x2	Selects function AD0.	
		0x3	Selects function CT32B0_MAT3.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
6	-	-	Reserved	1
7	ADMODE		Selects Analog/Digital mode	1
		0	Analog input mode	
		1	Digital functional mode	
9:8	-	-	Reserved	00
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.29 IOCON_R_PIO1_0

Table 125. IOCON_R_PIO1_0 register (IOCON_R_PIO1_0, address 0x4004 4078) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function R. This function is reserved. Select one of the alternate functions below.	
		0x1	Selects function PIO1_0.	
		0x2	Selects function AD1.	
		0x3	Selects function CT32B1_CAP0.	

Table 125. IOCON_R_PIO1_0 register (IOCON_R_PIO1_0, address 0x4004 4078) bit description

Bit	Symbol	Value	Description	Reset value
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
6	-	-	Reserved	1
7	ADMODE		Selects Analog/Digital mode	1
		0	Analog input mode	
		1	Digital functional mode	
9:8	-	-	Reserved	00
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.30 IOCON_R_PIO1_1

Table 126. IOCON_R_PIO1_1 register (IOCON_R_PIO1_1, address 0x4004 407C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function R. This function is reserved. Select one of the alternate functions below.	
		0x1	Selects function PIO1_1.	
		0x2	Selects function AD2.	
		0x3	Selects function CT32B1_MAT0.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
6	-	-	Reserved	1

Table 126. IOCON_R_PIO1_1 register (IOCON_R_PIO1_1, address 0x4004 407C) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
7	ADMODE		Selects Analog/Digital mode	1
		0	Analog input mode	
		1	Digital functional mode	
9:8	-	-	Reserved	00
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.31 IOCON_R_PIO1_2

Table 127. IOCON_R_PIO1_2 register (IOCON_R_PIO1_2, address 0x4004 4080) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function R. This function is reserved. Select one of the alternate functions below.	
		0x1	Selects function PIO1_2.	
		0x2	Selects function AD3.	
		0x3	Selects function CT32B1_MAT1.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
6	-	-	Reserved	1
7	ADMODE		Selects Analog/Digital mode	1
		0	Analog input mode	
		1	Digital functional mode	
9:8	-	-	Reserved	00
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.32 IOCON_PIO3_0

Table 128. IOCON_PIO3_0 register (IOCON_PIO3_0, address 0x4004 4084) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO3_0.	
		0x1	Selects function $\overline{\text{DTR}}$ (function not available on all parts).	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.33 IOCON_PIO3_1

Table 129. IOCON_PIO3_1 register (IOCON_PIO3_1, address 0x4004 4088) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO3_1.	
		0x1	Selects function $\overline{\text{DSR}}$ (function not available on all parts, must also be configured in the corresponding IOCON_DSR_LOC register).	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011

Table 129. IOCON_PIO3_1 register (IOCON_PIO3_1, address 0x4004 4088) bit description

Bit	Symbol	Value	Description	Reset value
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.34 IOCON_PIO2_3

Table 130. IOCON_PIO2_3 register (IOCON_PIO2_3, address 0x4004 408C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO2_3.	
		0x1	Selects function $\overline{\text{RI}}$.	
		0x2	Selects function MOSI1 (function not available on all parts).	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.35 IOCON_SWDIO_PIO1_3

Table 131. IOCON_SWDIO_PIO1_3 register (IOCON_SWDIO_PIO1_3, address 0x4004 4090) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function SWDIO.	
		0x1	Selects function PIO1_3.	
		0x2	Selects function AD4.	
		0x3	Selects function CT32B1_MAT2.	

Table 131. IOCON_SWDIO_PIO1_3 register (IOCON_SWDIO_PIO1_3, address 0x4004 4090) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
6	-	-	Reserved	1
7	ADMODE		Selects Analog/Digital mode	1
		0	Analog input mode	
		1	Digital functional mode	
9:8	-	-	Reserved	00
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.36 IOCON_PIO1_4

Table 132. IOCON_PIO1_4 register (IOCON_PIO1_4, address 0x4004 4094) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. This pin functions as WAKEUP pin if the LPC13xx is in Deep power-down mode regardless of the value of FUNC. All other values are reserved.	000
		0x0	Selects function PIO1_4.	
		0x1	Selects function AD5.	
		0x2	Selects function CT32B1_MAT3.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
6	-	-	Reserved	1

Table 132. IOCON_PIO1_4 register (IOCON_PIO1_4, address 0x4004 4094) bit description

Bit	Symbol	Value	Description	Reset value
7	ADMODE		Selects Analog/Digital mode	1
		0	Analog input mode	
		1	Digital functional mode	
9:8	-	-	Reserved	00
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.37 IOCON_PIO1_11

Table 133. IOCON_PIO1_11 register (IOCON_PIO1_11, address 0x4004 4098) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO1_11.	
		0x1	Selects function AD7.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
6	-	-	Reserved	1
7	ADMODE		Selects Analog/Digital mode	1
		0	Analog input mode	
		1	Digital functional mode	
9:8	-	-	Reserved	00
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.38 IOCON_PIO3_2

Table 134. IOCON_PIO3_2 register (IOCON_PIO3_2, address 0x4004 409C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO3_2.	
		0x1	Selects function $\overline{\text{DCD}}$ (function not available on all parts, must also be configured in the corresponding IOCON_DCD_LOC register).	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.39 IOCON_PIO1_5

Table 135. IOCON_PIO1_5 register (IOCON_PIO1_5, address 0x4004 40A0) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO1_5.	
		0x1	Selects function $\overline{\text{RTS}}$.	
		0x2	Selects function CT32B0_CAP0.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011

Table 135. IOCON_PIO1_5 register (IOCON_PIO1_5, address 0x4004 40A0) bit description

Bit	Symbol	Value	Description	Reset value
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.40 IOCON_PIO1_6

Table 136. IOCON_PIO1_6 register (IOCON_PIO1_6, address 0x4004 40A4) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO1_6.	
		0x1	Selects function RXD.	
		0x2	Selects function CT32B0_MAT0.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.41 IOCON_PIO1_7

Table 137. IOCON_PIO1_7 register (IOCON_PIO1_7, address 0x4004 40A8) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO1_7.	
		0x1	Selects function TXD.	
		0x2	Selects function CT32B0_MAT1.	

Table 137. IOCON_PIO1_7 register (IOCON_PIO1_7, address 0x4004 40A8) bit description

Bit	Symbol	Value	Description	Reset value
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.42 IOCON_PIO_3_3

Table 138. IOCON_PIO3_3 register (IOCON_PIO3_3, address 0x4004 40AC) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function PIO3_3.	
		0x1	Selects function $\overline{\text{RI}}$ (function not available on all parts, must also be configured in the corresponding IOCON_RI_LOC register).	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control)	10
		0x0	Inactive (no pull-down/pull-up resistor enabled)	
		0x1	Pull-down resistor enabled	
		0x2	Pull-up resistor enabled	
		0x3	Repeater mode	
5	HYS		Hysteresis	0
		0	Disable	
		1	Enable	
9:6	-	-	Reserved	0011
10	OD		Selects pseudo open-drain mode.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.43 IOCON_SCK0_LOC

This register is used to select a pin among three possible choices for the SSP0 SCK0 function.

Remark: Note that once the pin location has been selected, the function still must be set to SCK0 in the corresponding IOCONF registers for the SCK0 to be usable on that pin.

Table 139. IOCON SCK0 location register (IOCON_SCK0_LOC, address 0x4004 40B0) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SCKLOC		Selects pin location for SCK0 pin.	0
		0x0	Selects SCK0 function for pin SWCLK/PIO0_10/SCK0/CT16B0_MAT2 (see Table 121).	
		0x1	Selects SCK0 function for pin PIO2_11/SCK0 (see Table 123	
		0x2	Selects SCK0 function for pin PIO0_6/USB_CONNECT/SCK0 (see Table 114).	
		0x3	Reserved.	
31:2	-	-	Reserved	-

7.4.44 IOCON_DSR_LOC

Remark: For the LPC1311/01 and LPC1313/01 parts, the modem functions on pins PIO3_1 to PIO3_3 must be configured in the corresponding IOCONFIG registers and also in the IOCON_DSR_LOC, IOCON_DCD_LOC, and IOCON_RI_LOC registers (see Table 140 to Table 142).

Table 140. IOCON DSR location register (IOCON_DSR_LOC, address 0x4004 40B4) bit description

Bit	Symbol	Value	Description	Reset value
1:0	DSRLOC		Selects pin location for DSR0 pin (this register is only used for parts LPC1311/01 and LPC1313/01).	00
		0x0	Selects $\overline{\text{DSR}}$ function in pin location PIO2_1/$\overline{\text{DSR}}$/SCK1.	
		0x1	Selects $\overline{\text{DSR}}$ function in pin location PIO3_1/$\overline{\text{DSR}}$.	
		0x2	Reserved.	
		0x3	Reserved.	
31:2	-	-	Reserved.	-

7.4.45 IOCON_DCD_LOC

Remark: For the LPC1311/01 and LPC1313/01 parts, the modem functions on pins PIO3_1 to PIO3_3 must be configured in the corresponding IOCONFIG registers and also in the IOCON_DSR_LOC, IOCON_DCD_LOC, and IOCON_RI_LOC registers (see Table 140 to Table 142).

Table 141. IOCON DCD location register (IOCON_DCD_LOC, address 0x4004 40B8) bit description

Bit	Symbol	Value	Description	Reset value
1:0	DCDLOC		Selects pin location for DCD pin (this register is only used for parts LPC1311/01 and LPC1313/01).	00
		0x0	Selects $\overline{\text{DCD}}$ function in pin location PIO2_2/$\overline{\text{DCD}}$/MISO1.	
		0x1	Selects $\overline{\text{DCD}}$ function in pin location PIO3_2/$\overline{\text{DCD}}$.	
		0x2	Reserved.	
		0x3	Reserved.	
31:2	-	-	Reserved.	-

7.4.46 IOCON_RI_LOC

Remark: For the LPC1311/01 and LPC1313/01 parts, the modem functions on pins PIO3_1 to PIO3_3 must be configured in the corresponding IOCONFIG registers and also in the IOCON_DSR_LOC, IOCON_DCD_LOC, and IOCON_RI_LOC registers (see Table 140 to Table 142).

Table 142. IOCON RI location register (IOCON_RI_LOC, address 0x4004 40BC) bit description

Bit	Symbol	Value	Description	Reset value
1:0	RILOC		Selects pin location for RI pin (this register is only used for parts LPC1311/01 and LPC1313/01)	00
		0x0	Selects $\overline{\text{RI}}$ function in pin location PIO2_3/$\overline{\text{RI}}$/MOSI1.	
		0x1	Selects $\overline{\text{RI}}$ function in pin location PIO3_3/$\overline{\text{RI}}$.	
		0x2	Reserved.	
		0x3	Reserved.	
31:2	-	-	Reserved.	-

8.1 How to read this chapter

The LPC13xx parts are available in LQFP48 and HVQFN33 packages. The LPC1342/43 parts have dedicated USB pins and additional USB functions on some pins.

Table 143. LPC13xx pin configuration overview

Part	HVQFN33 package	Pin description	LQFP48 package	Pin description
LPC1311, LPC1311/01	Figure 13	Table 145	-	-
LPC1313, LPC1313/01	Figure 13	Table 145	Figure 12	Table 144
LPC1342	Figure 11	Table 145	-	-
LPC1343	Figure 11	Table 145	Figure 10	Table 144

8.2 LPC134x pin configuration

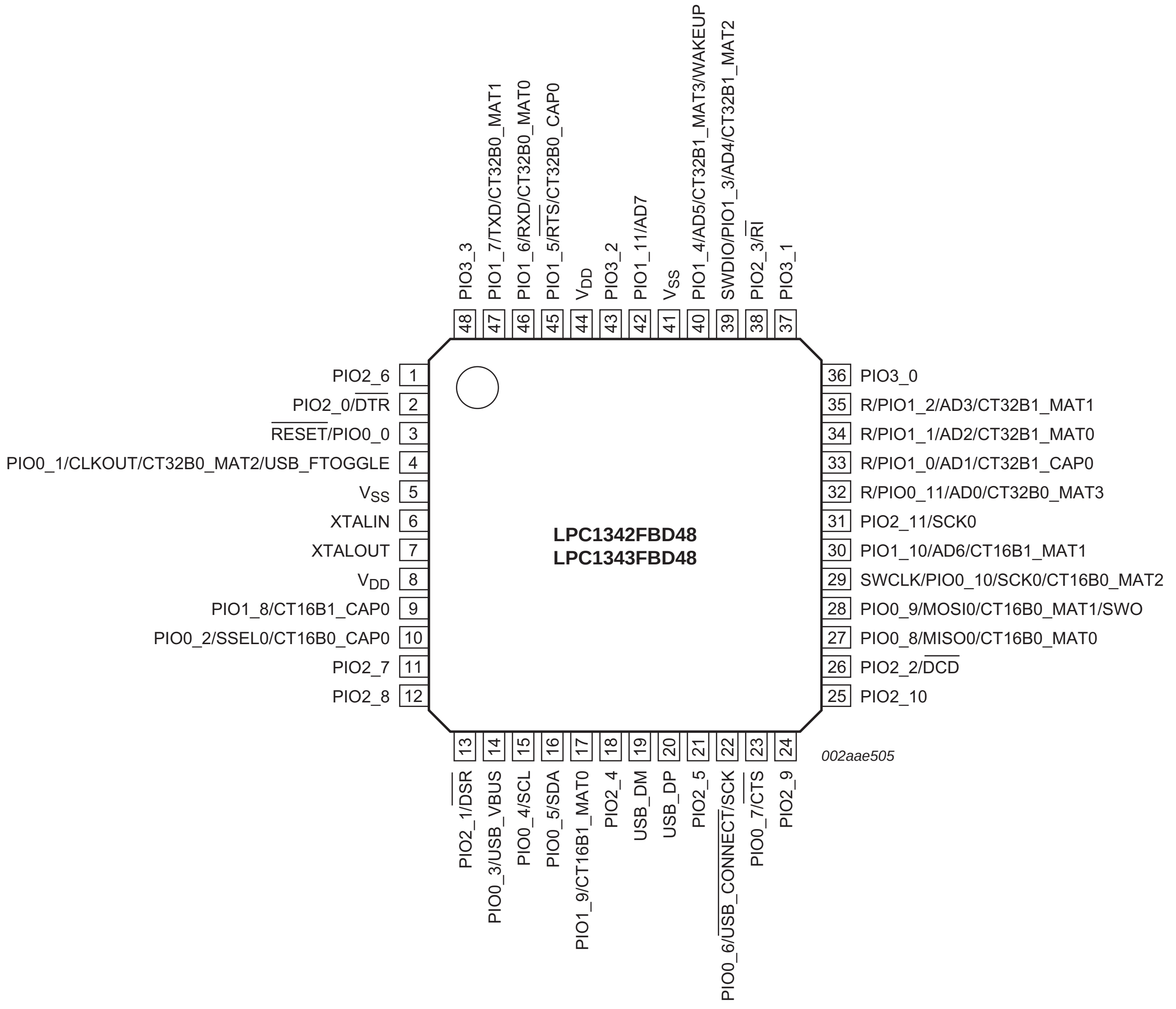


Fig 10. LPC1342/43 LQFP48 package

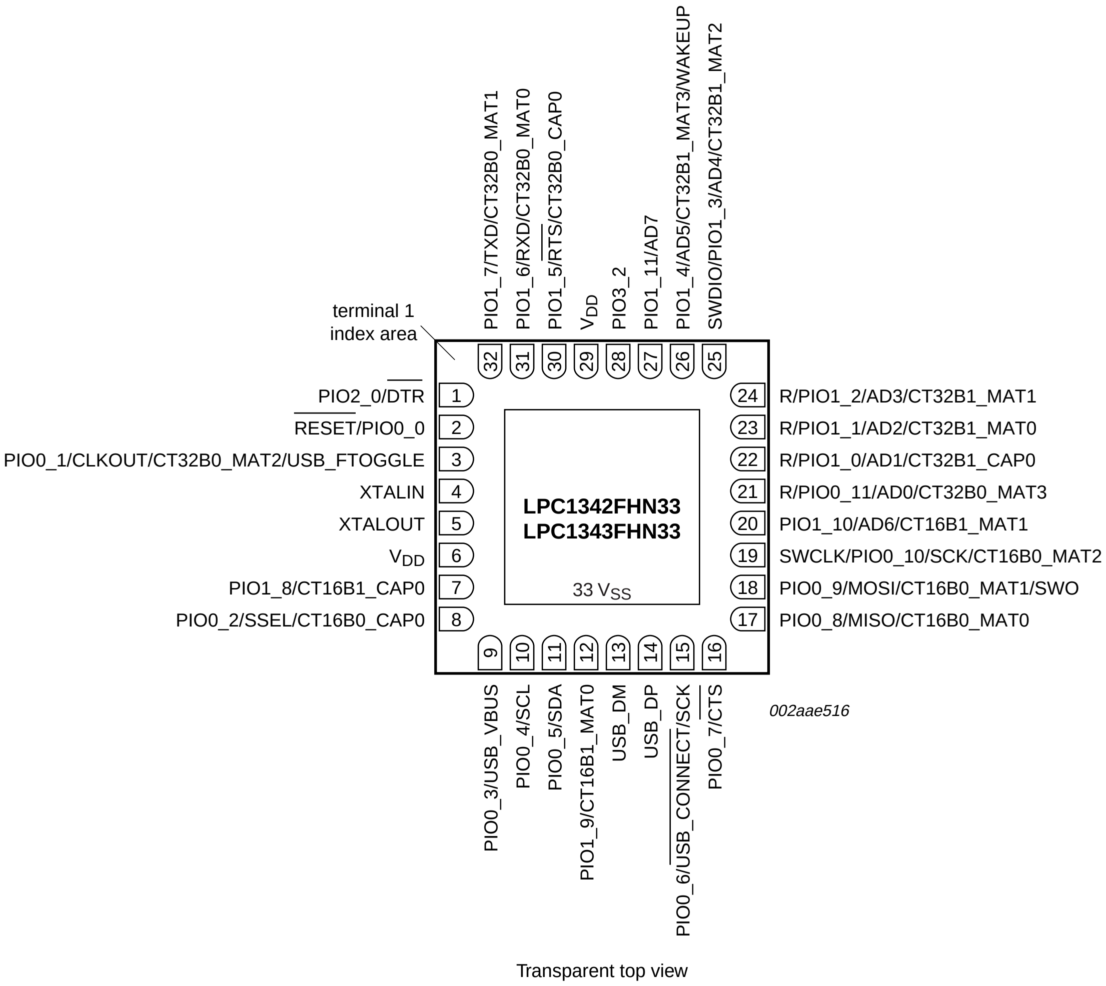
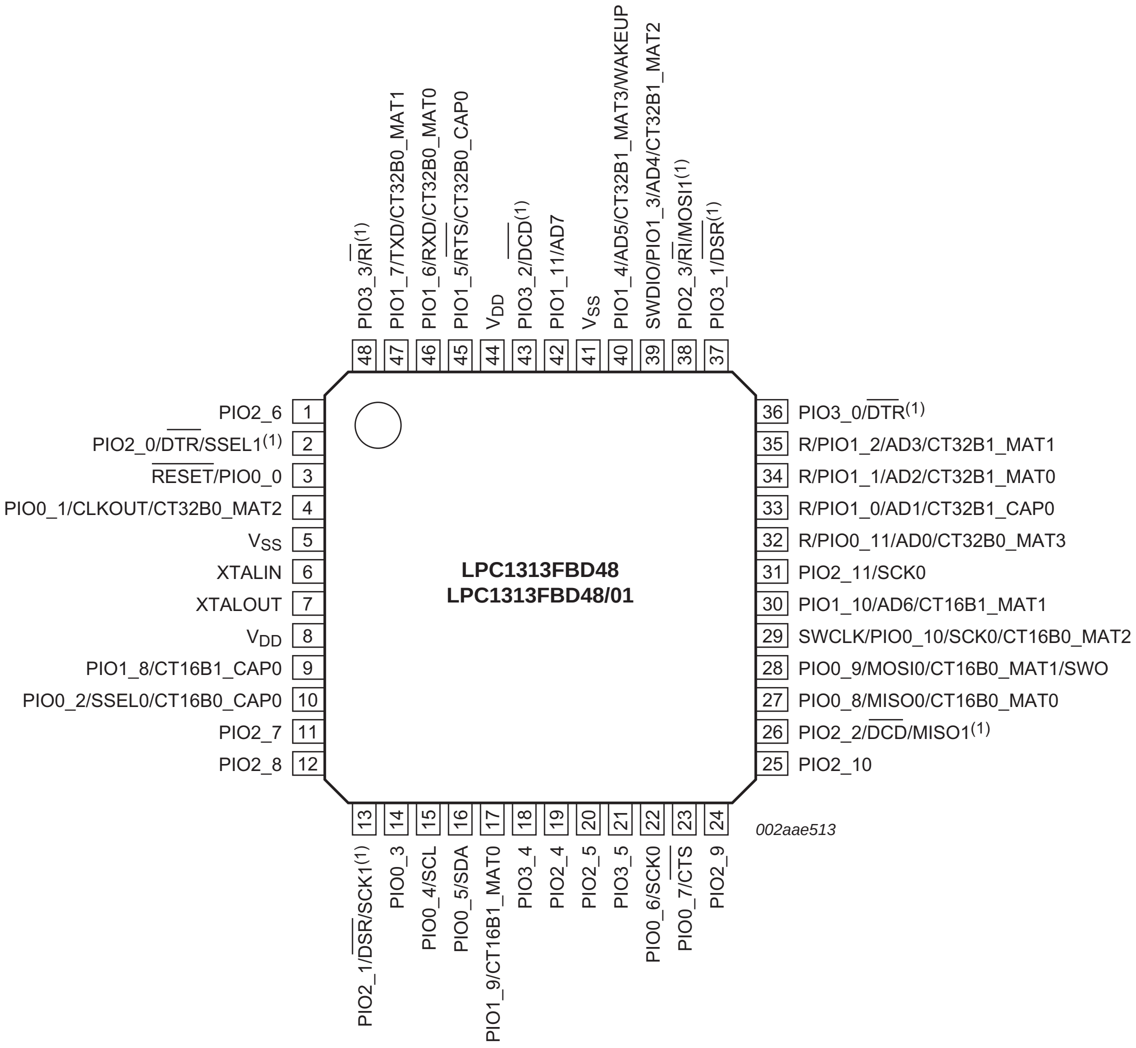


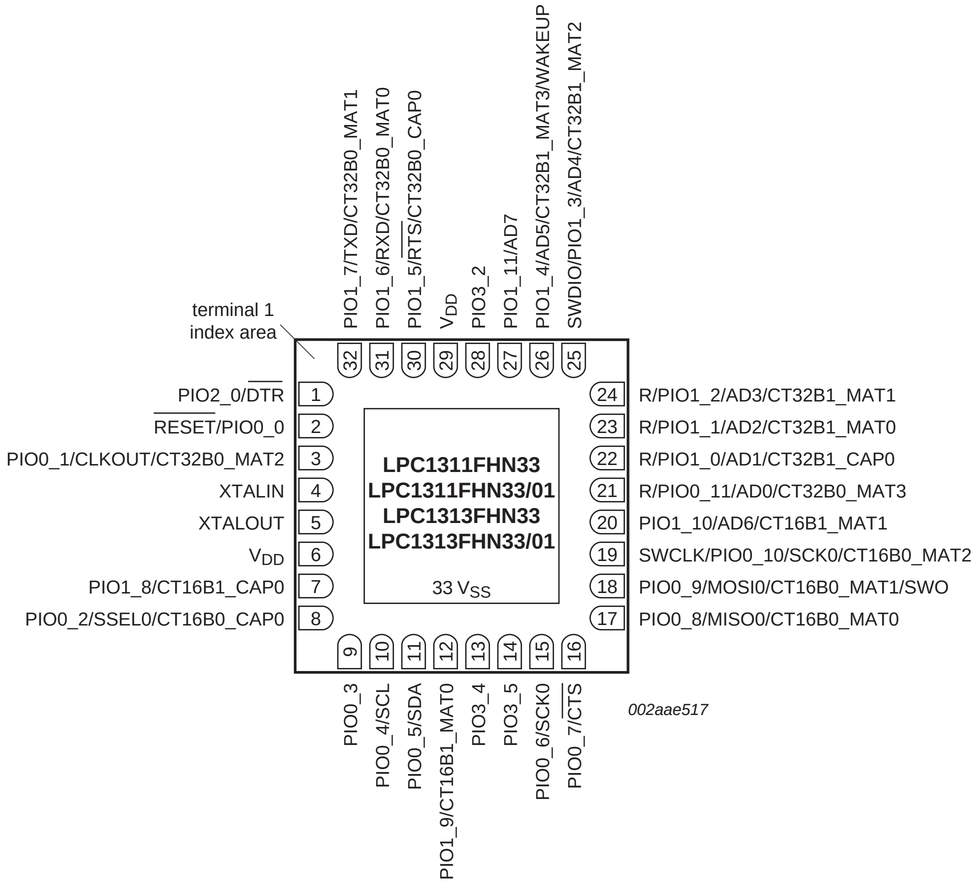
Fig 11. LPC1342/43 HVQFN33 package

8.3 LPC131x pin configuration



(1) SSP1 or UART function on LPC1313FBD48/01 only.

Fig 12. LPC1313 LQFP48 package



Transparent top view

Fig 13. LPC1311/13 HVQFN33 package

8.4 Pin description

In Table 144 and Table 145, the pins are listed in order of their port number. Supply pins and special function pins appear at the end.

The default function of each pin is always the first function listed in the description column or the first function of each pin symbol. Each pin function can be set through the corresponding IOCON register (see Table 96).

8.4.1 LQFP48 packages

Table 144. LPC1313/42/43 LQFP48 pin description table

Symbol	Pin	Start logic input	Type	Reset state [1]	Description
$\overline{\text{RESET}}$/PIO0_0	3[2]	yes	I	I; PU	**$\overline{\text{RESET}}$ —** External reset input with 20 ns glitch filter. A LOW-going pulse as short as 50 ns on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0.
			I/O	-	**PIO0_0 —** General purpose digital input/output pin with 10 ns glitch filter.
PIO0_1/CLKOUT/ CT32B0_MAT2/ USB_FTOGGLE	4[3]	yes	I/O	I; PU	**PIO0_1 —** General purpose digital input/output pin. A LOW level on this pin during reset starts the ISP command handler or the USB device enumeration (USB on LPC1343 only, see description of PIO0_3).
			O	-	**CLKOUT —** Clockout pin.
			O	-	**CT32B0_MAT2 —** Match output 2 for 32-bit timer 0.
			O	-	**USB_FTOGGLE —** USB 1 ms Start-of-Frame signal (LPC1343 only).
PIO0_2/SSEL0/ CT16B0_CAP0	10[3]	yes	I/O	I; PU	**PIO0_2 —** General purpose digital input/output pin.
			I/O	-	**SSEL0 —** Slave select for SSP0.
			I	-	**CT16B0_CAP0 —** Capture input 0 for 16-bit timer 0.
PIO0_3/USB_VBUS	14[3]	yes	I/O	I; PU	**PIO0_3 —** General purpose digital input/output pin. LPC1343 only: A LOW level on this pin during reset starts the ISP command handler, a HIGH level starts the USB device enumeration.
			I	-	**USB_VBUS —** Monitors the presence of USB bus power (LPC1343 only).
PIO0_4/SCL	15[4]	yes	I/O	I; IA	**PIO0_4 —** General purpose digital input/output pin (open-drain).
			I/O	-	**SCL —** I^{2}C-bus clock input/output (open-drain). High-current sink only if I^{2}C Fast-mode Plus is selected in the I/O configuration register.
PIO0_5/SDA	16[4]	yes	I/O	I; IA	**PIO0_5 —** General purpose digital input/output pin (open-drain).
			I/O	-	**SDA —** I^{2}C-bus data input/output (open-drain). High-current sink only if I^{2}C Fast-mode Plus is selected in the I/O configuration register.
PIO0_6/ $\overline{\text{USB\_CONNECT}}$/ SCK0	22[3]	yes	I/O	I; PU	**PIO0_6 —** General purpose digital input/output pin.
			O	-	**$\overline{\text{USB\_CONNECT}}$ —** Signal used to switch an external 1.5 kΩ resistor under software control. Used with the SoftConnect USB feature (LPC1343 only).
			I/O	-	**SCK0 —** Serial clock for SSP0.
PIO0_7/$\overline{\text{CTS}}$	23[3]	yes	I/O	I; PU	**PIO0_7 —** General purpose digital input/output pin (high-current output driver).
			I	-	**$\overline{\text{CTS}}$ —** Clear To Send input for UART.
PIO0_8/MISO0/ CT16B0_MAT0	27[3]	yes	I/O	I; PU	**PIO0_8 —** General purpose digital input/output pin.
			I/O	-	**MISO0 —** Master In Slave Out for SSP0.
			O	-	**CT16B0_MAT0 —** Match output 0 for 16-bit timer 0.

Table 144. LPC1313/42/43 LQFP48 pin description table *…continued*

Symbol	Pin	Start logic input	Type	Reset state [1]	Description
PIO0_9/MOSI0/ CT16B0_MAT1/ SWO	28[3]	yes	I/O	I; PU	**PIO0_9 —** General purpose digital input/output pin.
			I/O	-	**MOSI0 —** Master Out Slave In for SSP0.
			O	-	**CT16B0_MAT1 —** Match output 1 for 16-bit timer 0.
			O	-	**SWO —** Serial wire trace output.
SWCLK/PIO0_10/ SCK0/CT16B0_MAT2	29[3]	yes	I	I; PU	**SWCLK —** Serial wire clock.
			I/O	-	**PIO0_10 —** General purpose digital input/output pin.
			I/O	-	**SCK0 —** Serial clock for SSP0.
			O	-	**CT16B0_MAT2 —** Match output 2 for 16-bit timer 0.
R/PIO0_11/ AD0/CT32B0_MAT3	32[5]	yes	-	I; PU	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
			I/O	-	**PIO0_11 —** General purpose digital input/output pin.
			I	-	**AD0 —** A/D converter, input 0.
			O	-	**CT32B0_MAT3 —** Match output 3 for 32-bit timer 0.
R/PIO1_0/ AD1/CT32B1_CAP0	33[5]	yes	-	I; PU	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
			I/O	-	**PIO1_0 —** General purpose digital input/output pin.
			I	-	**AD1 —** A/D converter, input 1.
			I	-	**CT32B1_CAP0 —** Capture input 0 for 32-bit timer 1.
R/PIO1_1/ AD2/CT32B1_MAT0	34[5]	yes	-	I; PU	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
			I/O	-	**PIO1_1 —** General purpose digital input/output pin.
			I	-	**AD2 —** A/D converter, input 2.
			O	-	**CT32B1_MAT0 —** Match output 0 for 32-bit timer 1.
R/PIO1_2/ AD3/CT32B1_MAT1	35[5]	yes	-	I; PU	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
			I/O	-	**PIO1_2 —** General purpose digital input/output pin.
			I	-	**AD3 —** A/D converter, input 3.
			O	-	**CT32B1_MAT1 —** Match output 1 for 32-bit timer 1.
SWDIO/PIO1_3/ AD4/ CT32B1_MAT2	39[5]	yes	I/O	I; PU	**SWDIO —** Serial wire debug input/output.
			I/O	-	**PIO1_3 —** General purpose digital input/output pin.
			I	-	**AD4 —** A/D converter, input 4.
			O	-	**CT32B1_MAT2 —** Match output 2 for 32-bit timer 1.
PIO1_4/AD5/ CT32B1_MAT3/ WAKEUP	40[5]	yes	I/O	I; PU	**PIO1_4 —** General purpose digital input/output pin.
			I	-	**AD5 —** A/D converter, input 5.
			O	-	**CT32B1_MAT3 —** Match output 3 for 32-bit timer 1.
			I	-	**WAKEUP —** Deep power-down mode wake-up pin with 20 ns glitch filter. This pin must be pulled HIGH externally to enter Deep power-down mode and pulled LOW to exit Deep power-down mode. A LOW-going pulse as short as 50 ns wakes up the part.

Table 144. LPC1313/42/43 LQFP48 pin description table *…continued*

Symbol	Pin	Start logic input	Type	Reset state [1]	Description
PIO1_5/$\overline{\text{RTS}}$/CT32B0_CAP0	45[3]	yes	I/O	I; PU	**PIO1_5** — General purpose digital input/output pin.
			O	-	**$\overline{\text{RTS}}$** — Request To Send output for UART.
			I	-	**CT32B0_CAP0** — Capture input 0 for 32-bit timer 0.
PIO1_6/RXD/CT32B0_MAT0	46[3]	yes	I/O	I; PU	**PIO1_6** — General purpose digital input/output pin.
			I	-	**RXD** — Receiver input for UART.
			O	-	**CT32B0_MAT0** — Match output 0 for 32-bit timer 0.
PIO1_7/TXD/CT32B0_MAT1	47[3]	yes	I/O	I; PU	**PIO1_7** — General purpose digital input/output pin.
			O	-	**TXD** — Transmitter output for UART.
			O	-	**CT32B0_MAT1** — Match output 1 for 32-bit timer 0.
PIO1_8/CT16B1_CAP0	9[3]	yes	I/O	I; PU	**PIO1_8** — General purpose digital input/output pin.
			I	-	**CT16B1_CAP0** — Capture input 0 for 16-bit timer 1.
PIO1_9/CT16B1_MAT0	17[3]	yes	I/O	I; PU	**PIO1_9** — General purpose digital input/output pin.
			O	-	**CT16B1_MAT0** — Match output 0 for 16-bit timer 1.
PIO1_10/AD6/CT16B1_MAT1	30[5]	yes	I/O	I; PU	**PIO1_10** — General purpose digital input/output pin.
			I	-	**AD6** — A/D converter, input 6.
			O	-	**CT16B1_MAT1** — Match output 1 for 16-bit timer 1.
PIO1_11/AD7	42[5]	yes	I/O	I; PU	**PIO1_11** — General purpose digital input/output pin.
			I	-	**AD7** — A/D converter, input 7.
PIO2_0/$\overline{\text{DTR}}$/SSEL1	2[3]	yes	I/O	I; PU	**PIO2_0** — General purpose digital input/output pin.
			O	-	**$\overline{\text{DTR}}$** — Data Terminal Ready output for UART.
			I/O	-	**SSEL1** — Slave Select for SSP1 (LPC1313FBD48/01 only).
PIO2_1/$\overline{\text{DSR}}$/SCK1	13[3]	yes	I/O	I; PU	**PIO2_1** — General purpose digital input/output pin.
			I	-	**$\overline{\text{DSR}}$** — Data Set Ready input for UART.
			I/O	-	**SCK1** — Serial clock for SSP1 (LPC1313FBD48/01 only).
PIO2_2/$\overline{\text{DCD}}$/MISO1	26[3]	yes	I/O	I; PU	**PIO2_2** — General purpose digital input/output pin.
			I	-	**$\overline{\text{DCD}}$** — Data Carrier Detect input for UART.
			I/O	-	**MISO1** — Master In Slave Out for SSP1 (LPC1313FBD48/01 only).
PIO2_3/$\overline{\text{RI}}$/MOSI1	38[3]	yes	I/O	I; PU	**PIO2_3** — General purpose digital input/output pin.
			I	-	**$\overline{\text{RI}}$** — Ring Indicator input for UART.
			I/O	-	**MOSI1** — Master Out Slave In for SSP1 (LPC1313FBD48/01 only).
PIO2_4	18[3]	yes	I/O	I; PU	**PIO2_4** — General purpose digital input/output pin (LPC1343 only).
PIO2_4	19[3]	yes	I/O	I; PU	**PIO2_4** — General purpose digital input/output pin (LPC1313 only).
PIO2_5	21[3]	yes	I/O	I; PU	**PIO2_5** — General purpose digital input/output pin (LPC1343 only).
PIO2_5	20[3]	yes	I/O	I; PU	**PIO2_5** — General purpose digital input/output pin (LPC1313 only).
PIO2_6	1[3]	yes	I/O	I; PU	**PIO2_6** — General purpose digital input/output pin.
PIO2_7	11[3]	yes	I/O	I; PU	**PIO2_7** — General purpose digital input/output pin.
PIO2_8	12[3]	yes	I/O	I; PU	**PIO2_8** — General purpose digital input/output pin.
PIO2_9	24[3]	yes	I/O	I; PU	**PIO2_9** — General purpose digital input/output pin.
PIO2_10	25[3]	yes	I/O	I; PU	**PIO2_10** — General purpose digital input/output pin.

Table 144. LPC1313/42/43 LQFP48 pin description table *…continued*

Symbol	Pin	Start logic input	Type	Reset state [1]	Description
PIO2_11/SCK0	31[3]	yes	I/O	I; PU	**PIO2_11 —** General purpose digital input/output pin.
			I/O	-	**SCK0 —** Serial clock for SSP0.
PIO3_0/$\overline{\text{DTR}}$	36[3]	yes	I/O	I; PU	**PIO3_0 —** General purpose digital input/output pin.
			O	-	**$\overline{\text{DTR}}$ —** Data Terminal Ready output for UART (LPC1311/01 and LPC1313/01 only).
PIO3_1/$\overline{\text{DSR}}$	37[3]	yes	I/O	I; PU	**PIO3_1 —** General purpose digital input/output pin.
			I	-	**$\overline{\text{DSR}}$ —** Data Set Ready input for UART (LPC1311/01 and LPC1313/01 only).
PIO3_2/$\overline{\text{DCD}}$	43[3]	yes	I/O	I; PU	**PIO3_2 —** General purpose digital input/output pin.
			I	-	**$\overline{\text{DCD}}$ —** Data Carrier Detect input for UART (LPC1311/01 and LPC1313/01 only).
PIO3_3/$\overline{\text{RI}}$	48[3]	yes	I/O	I; PU	**PIO3_3 —** General purpose digital input/output pin.
			I	-	**$\overline{\text{RI}}$ —** Ring Indicator input for UART (LPC1311/01 and LPC1313/01 only).
PIO3_4	18[3]	no	I/O	I; PU	**PIO3_4 —** General purpose digital input/output pin (LPC1313 only).
PIO3_5	21[3]	no	I/O	I; PU	**PIO3_5 —** General purpose digital input/output pin (LPC1313 only).
USB_DM	19[6]	no	I/O	F	**USB_DM —** USB bidirectional D– line (LPC1343 only).
USB_DP	20[6]	no	I/O	F	**USB_DP —** USB bidirectional D+ line (LPC1343 only).
V_{DD}	8; 44	-	I	-	3.3 V supply voltage to the internal regulator, the external rail, and the ADC. Also used as the ADC reference voltage.
XTALIN	6[7]	-	I	-	Input to the oscillator circuit and internal clock generator circuits. Input voltage must not exceed 1.8 V.
XTALOUT	7[7]	-	O	-	Output from the oscillator amplifier.
V_{SS}	5; 41	-	I	-	Ground.

[1] Pin state at reset for default function: I = Input; O = Output; PU = internal pull-up enabled (for V_{DD} = 3.3 V, pin is pulled up to 2.6 V for parts LPC1311/13/42/43 and pulled up to 3.3 V for parts LPC1311/01 and LPC1313/01); IA = inactive, no pull-up/down enabled; F = floating; floating pins, if not used, should be tied to ground or power to minimize power consumption.

[2] 5 V tolerant pad. See the LPC13xx data sheet for pad characteristics. $\overline{\text{RESET}}$ functionality is not available in Deep power-down mode. Use the WAKEUP pin to reset the chip and wake up from Deep power-down mode. An external pull-up resistor is required on this pin for the Deep power-down mode.

[3] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors and configurable hysteresis (see Figure 9).

[4] I^2C-bus pads compliant with the I^2C-bus specification for I^2C standard mode and I^2C Fast-mode Plus.

[5] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors, configurable hysteresis, and analog input. When configured as a ADC input, digital section of the pad is disabled and the pin is not 5 V tolerant (see Figure 9).

[6] Pad provides USB functions. It is designed in accordance with the USB specification, revision 2.0 (Full-speed and Low-speed mode only). This pad is not 5 V tolerant.

[7] When the system oscillator is not used, connect XTALIN and XTALOUT as follows: XTALIN can be left floating or can be grounded (grounding is preferred to reduce susceptibility to noise). XTALOUT should be left floating.

8.4.2 HVQFN33 packages

Table 145. LPC1311/13/42/43 HVQFN33 pin description table

Symbol	Pin	Start logic input	Type	Reset state [1]	Description
$\overline{\text{RESET}}$/PIO0_0	2[2]	yes	I	I; PU	**$\overline{\text{RESET}}$** — External reset input with 20 ns glitch filter. A LOW-going pulse as short as 50 ns on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0.
			I/O	-	**PIO0_0** — General purpose digital input/output pin with 10 ns glitch filter.
PIO0_1/CLKOUT/ CT32B0_MAT2/ USB_FTOGGLE	3[3]	yes	I/O	I; PU	**PIO0_1** — General purpose digital input/output pin. A LOW level on this pin during reset starts the ISP command handler or the USB device enumeration (USB on LPC1342/43 only, see description of PIO0_3).
			O	-	**CLKOUT** — Clock out pin.
			O	-	**CT32B0_MAT2** — Match output 2 for 32-bit timer 0.
			O	-	**USB_FTOGGLE** — USB 1 ms Start-of-Frame signal (LPC1342/43 only).
PIO0_2/SSEL0/ CT16B0_CAP0	8[3]	yes	I/O	I; PU	**PIO0_2** — General purpose digital input/output pin.
			I/O	-	**SSEL0** — Slave select for SSP0.
			I	-	**CT16B0_CAP0** — Capture input 0 for 16-bit timer 0.
PIO0_3/ USB_VBUS	9[3]	yes	I/O	I; PU	**PIO0_3** — General purpose digital input/output pin. LPC1342/43 only: A LOW level on this pin during reset starts the ISP command handler, a HIGH level starts the USB device enumeration.
			I	-	**USB_VBUS** — Monitors the presence of USB bus power (LPC1342/43 only).
PIO0_4/SCL	10[4]	yes	I/O	I; IA	**PIO0_4** — General purpose digital input/output pin (open-drain).
			I/O	-	**SCL** — I^2C-bus clock input/output (open-drain). High-current sink only if I^2C Fast-mode Plus is selected in the I/O configuration register.
PIO0_5/SDA	11[4]	yes	I/O	I; IA	**PIO0_5** — General purpose digital input/output pin (open-drain).
			I/O	-	**SDA** — I^2C-bus data input/output (open-drain). High-current sink only if I^2C Fast-mode Plus is selected in the I/O configuration register.
PIO0_6/ $\overline{\text{USB\_CONNECT}}$/ SCK0	15[3]	yes	I/O	I; PU	**PIO0_6** — General purpose digital input/output pin.
			O	-	**$\overline{\text{USB\_CONNECT}}$** — Signal used to switch an external 1.5 kΩ resistor under software control. Used with the SoftConnect USB feature (LPC1342/43 only).
			I/O	-	**SCK0** — Serial clock for SSP0.
PIO0_7/$\overline{\text{CTS}}$	16[3]	yes	I/O	I; PU	**PIO0_7** — General purpose digital input/output pin (high-current output driver).
			I	-	**$\overline{\text{CTS}}$** — Clear To Send input for UART.
PIO0_8/MISO0/ CT16B0_MAT0	17[3]	yes	I/O	I; PU	**PIO0_8** — General purpose digital input/output pin.
			I/O	-	**MISO0** — Master In Slave Out for SSP0.
			O	-	**CT16B0_MAT0** — Match output 0 for 16-bit timer 0.
PIO0_9/MOSI0/ CT16B0_MAT1/ SWO	18[3]	yes	I/O	I; PU	**PIO0_9** — General purpose digital input/output pin.
			I/O	-	**MOSI0** — Master Out Slave In for SSP0.
			O	-	**CT16B0_MAT1** — Match output 1 for 16-bit timer 0.
			O	-	**SWO** — Serial wire trace output.

All information provided in this document is subject to legal disclaimers.

Table 145. LPC1311/13/42/43 HVQFN33 pin description table *…continued*

Symbol	Pin	Start logic input	Type	Reset state [1]	Description
SWCLK/PIO0_10/ SCK0/ CT16B0_MAT2	19[3]	yes	I	I; PU	**SWCLK —** Serial wire clock.
			I/O	-	**PIO0_10 —** General purpose digital input/output pin.
			I/O	-	**SCK0 —** Serial clock for SSP0.
			O	-	**CT16B0_MAT2 —** Match output 2 for 16-bit timer 0.
R/PIO0_11/AD0/ CT32B0_MAT3	21[5]	yes	-	I; PU	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
			I/O	-	**PIO0_11 —** General purpose digital input/output pin.
			I	-	**AD0 —** A/D converter, input 0.
			O	-	**CT32B0_MAT3 —** Match output 3 for 32-bit timer 0.
R/PIO1_0/AD1/ CT32B1_CAP0	22[5]	yes	-	I; PU	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
			I/O	-	**PIO1_0 —** General purpose digital input/output pin.
			I	-	**AD1 —** A/D converter, input 1.
			I	-	**CT32B1_CAP0 —** Capture input 0 for 32-bit timer 1.
R/PIO1_1/AD2/ CT32B1_MAT0	23[5]	yes	-	I; PU	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
			I/O	-	**PIO1_1 —** General purpose digital input/output pin.
			I	-	**AD2 —** A/D converter, input 2.
			O	-	**CT32B1_MAT0 —** Match output 0 for 32-bit timer 1.
R/PIO1_2/AD3/ CT32B1_MAT1	24[5]	yes	-	I; PU	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
			I/O	-	**PIO1_2 —** General purpose digital input/output pin.
			I	-	**AD3 —** A/D converter, input 3.
			O	-	**CT32B1_MAT1 —** Match output 1 for 32-bit timer 1.
SWDIO/PIO1_3/ AD4/ CT32B1_MAT2	25[5]	yes	I/O	I; PU	**SWDIO —** Serial wire debug input/output.
			I/O	-	**PIO1_3 —** General purpose digital input/output pin.
			I	-	**AD4 —** A/D converter, input 4.
			O	-	**CT32B1_MAT2 —** Match output 2 for 32-bit timer 1.
PIO1_4/AD5/ CT32B1_MAT3/ WAKEUP	26[5]	yes	I/O	I; PU	**PIO1_4 —** General purpose digital input/output pin.
			I	-	**AD5 —** A/D converter, input 5.
			O	-	**CT32B1_MAT3 —** Match output 3 for 32-bit timer 1.
			I	-	**WAKEUP —** Deep power-down mode wake-up pin with 20 ns glitch filter. This pin must be pulled HIGH externally to enter Deep power-down mode and pulled LOW to exit Deep power-down mode. A LOW-going pulse as short as 50 ns wakes up the part.
PIO1_5/$\overline{\text{RTS}}$/ CT32B0_CAP0	30[3]	yes	I/O	I; PU	**PIO1_5 —** General purpose digital input/output pin.
			O	-	**$\overline{\text{RTS}}$ —** Request To Send output for UART.
			I	-	**CT32B0_CAP0 —** Capture input 0 for 32-bit timer 0.
PIO1_6/RXD/ CT32B0_MAT0	31[3]	yes	I/O	I; PU	**PIO1_6 —** General purpose digital input/output pin.
			I	-	**RXD —** Receiver input for UART.
			O	-	**CT32B0_MAT0 —** Match output 0 for 32-bit timer 0.

Table 145. LPC1311/13/42/43 HVQFN33 pin description table *…continued*

Symbol	Pin	Start logic input	Type	Reset state [1]	Description
PIO1_7/TXD/ CT32B0_MAT1	32[3]	yes	I/O	I; PU	**PIO1_7 —** General purpose digital input/output pin.
			O	-	**TXD —** Transmitter output for UART.
			O	-	**CT32B0_MAT1 —** Match output 1 for 32-bit timer 0.
PIO1_8/ CT16B1_CAP0	7[3]	yes	I/O	I; PU	**PIO1_8 —** General purpose digital input/output pin.
			I	-	**CT16B1_CAP0 —** Capture input 0 for 16-bit timer 1.
PIO1_9/ CT16B1_MAT0	12[3]	yes	I/O	I; PU	**PIO1_9 —** General purpose digital input/output pin.
			O	-	**CT16B1_MAT0 —** Match output 0 for 16-bit timer 1.
PIO1_10/AD6/ CT16B1_MAT1	20[5]	yes	I/O	I; PU	**PIO1_10 —** General purpose digital input/output pin.
			I	-	**AD6 —** A/D converter, input 6.
			O	-	**CT16B1_MAT1 —** Match output 1 for 16-bit timer 1.
PIO1_11/AD7	27[5]	yes	I/O	I; PU	**PIO1_11 —** General purpose digital input/output pin.
			I	-	**AD7 —** A/D converter, input 7.
PIO2_0/$\overline{\text{DTR}}$	1[3]	yes	I/O	I; PU	**PIO2_0 —** General purpose digital input/output pin.
			O	-	**$\overline{\text{DTR}}$ —** Data Terminal Ready output for UART.
PIO3_2	28[3]	yes	I/O	I; PU	**PIO3_2 —** General purpose digital input/output pin.
PIO3_4	13[3]	no	I/O	I; PU	**PIO3_4 —** General purpose digital input/output pin (LPC1311/13 only).
PIO3_5	14[3]	no	I/O	I; PU	**PIO3_5 —** General purpose digital input/output pin (LPC1311/13 only).
USB_DM	13[6]	no	I/O	F	**USB_DM —** USB bidirectional D– line (LPC1342/43 only).
USB_DP	14[6]	no	I/O	F	**USB_DP —** USB bidirectional D+ line (LPC1342/43 only).
V_{DD}	6; 29	-	I	-	3.3 V supply voltage to the internal regulator, the external rail, and the ADC. Also used as the ADC reference voltage.
XTALIN	4[7]	-	I	-	Input to the oscillator circuit and internal clock generator circuits. Input voltage must not exceed 1.8 V.
XTALOUT	5[7]	-	O	-	Output from the oscillator amplifier.
V_{SS}	33	-	-	-	Thermal pad. Connect to ground.

[1] Pin state at reset for default function: I = Input; O = Output; PU = internal pull-up enabled (for V_{DD} = 3.3 V, pin is pulled up to 2.6 V for parts LPC1311/13/42/43 and pulled up to 3.3 V for parts LPC1311/01 and LPC1313/01); IA = inactive, no pull-up/down enabled. F = floating; floating pins, if not used, should be tied to ground or power to minimize power consumption.

[2] 5 V tolerant pad. See LPC13xx data sheet for pad characteristics. $\overline{\text{RESET}}$ functionality is not available in Deep power-down mode. Use the WAKEUP pin to reset the chip and wake up from Deep power-down mode. An external pull-up resistor is required on this pin for the Deep power-down mode.

[3] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors and configurable hysteresis (see Figure 9).

[4] I^2C-bus pads compliant with the I^2C-bus specification for I^2C standard mode and I^2C Fast-mode Plus.

[5] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors, configurable hysteresis, and analog input. When configured as a ADC input, digital section of the pad is disabled, and the pin is not 5 V tolerant (see Figure 9).

[6] Pad provides USB functions. It is designed in accordance with the USB specification, revision 2.0 (Full-speed and Low-speed mode only). This pad is not 5 V tolerant.

[7] When the system oscillator is not used, connect XTALIN and XTALOUT as follows: XTALIN can be left floating or can be grounded (grounding is preferred to reduce susceptibility to noise). XTALOUT should be left floating.

UM10375

Chapter 9: LPC13xx General Purpose I/O (GPIO)

Rev. 5 — 21 June 2012 User manual

9.1 How to read this chapter

The number of GPIO pins available on each port depends on the LPC13xx part and the package. See Table 146 for available GPIO pins:

Table 146. GPIO configuration

Part	Package	GPIO port 0	GPIO port 1	GPIO port 2	GPIO port 3	Total GPIO pins
LPC1311, LPC1311/01	HVQFN33	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0	PIO3_2; PIO3_4; PIO3_5	28
LPC1313, LPC1313/01	LQFP48	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0 to PIO2_11	PIO3_0 to PIO3_5	42
LPC1313	HVQFN33	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0	PIO3_2; PIO3_4; PIO3_5	28
LPC1342	LQFP48	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0 to PIO2_11	PIO3_0 to PIO3_3	40
	HVQFN33	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0	PIO3_2	26
LPC1343	LQFP48	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0 to PIO2_11	PIO3_0 to PIO3_3	40
	HVQFN33	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0	PIO3_2	26

9.2 Features

- GPIO pins can be configured as input or output by software.
- Each individual port pin can serve as an edge- or level-sensitive interrupt request.
- Interrupts can be configured on single falling or rising edges and on both edges.
- Level-sensitive interrupt pins can be HIGH- or LOW-active.
- All GPIO pins are inputs by default.
- Reading and writing of data registers are masked by address bits 13:2.

9.3 Pin description

Table 147. GPIO pin description[1]

Pin	Type	Description
PIO0_0 to PIO0_11	I/O	GPIO port 0 input/output pins.
PIO1_0 to PIO1_11	I/O	GPIO port 1 input/output pins.
PIO2_0 to PIO2_11	I/O	GPIO port 2 input/output pins.
PIO3_0 to PIO3_11	I/O	GPIO port 3 input/output pins.

[1] The pin configuration depends on the LPC13xx package (see Table 146).

9.4 Register description

Each GPIO register can be up to 12 bits wide and can be read or written using word or half-word operations at word addresses.

Table 148. Register overview: GPIO (base address port 0: 0x5000 0000; port 1: 0x5001 0000, port 2: 0x5002 0000; port 3: 0x5003 0000)

Name	Access	Address offset	Description	Reset value
GPIODATAMASK	R/W	0x0000 to 0x3FF8	Port n data address masking register locations for pins PIOn_0 to PIOn_11 (see Section 9.5.1)	n/a
GPIODATA	R/W	0x3FFC	Port n data register for pins PIOn_0 to PIOn_11	n/a
-	-	0x4000 to 0x7FFC	Reserved	-
GPIODIR	R/W	0x8000	Data direction register for port n	0x00
GPIOIS	R/W	0x8004	Interrupt sense register for port n	0x00
GPIOIBE	R/W	0x8008	Interrupt both edges register for port n	0x00
GPIOIEV	R/W	0x800C	Interrupt event register for port n	0x00
GPIOIE	R/W	0x8010	Interrupt mask register for port n	0x00
GPIORIS	R	0x8014	Raw interrupt status register for port n	0x00
GPIOMIS	R	0x8018	Masked interrupt status register for port n	0x00
GPIOIC	W	0x801C	Interrupt clear register for port n	0x00
-	-	0x8020 - 0xFFFF	Reserved	0x00

9.4.1 GPIO data register

The GPIODATA register holds the current state of the pin (HIGH or LOW), independently of whether the pin is configured as an GPIO input or output or as another digital function. If the pin is configured as GPIO output, the current value of the GPIODATA register is driven to the pin.

Table 149. GPIO data register (GPIO0DATA, address 0x5000 3FFC; GPIO1DATA, address 0x5001 3FFC; GPIO2DATA, address 0x5002 3FFC; GPIO3DATA, address 0x5003 3FFC) bit description

Bit	Symbol	Description	Reset value	Access
11:0	DATA	Logic levels for pins PIOn_0 to PIOn_11. HIGH = 1, LOW = 0.	n/a	R/W
31:12	-	Reserved	0x00	-

A read of the GPIODATA register always returns the current logic level (state) of the pin independently of its configuration. Because there is a single data register for both the value of the output driver and the state of the pin’s input, write operations have different effects depending on the pin’s configuration:

- If a pin is configured as GPIO input, a write to the GPIODATA register has no effect on the pin level. A read returns the current state of the pin.

- If a pin is configured as GPIO output, the current value of GPIODATA register is driven to the pin. This value can be a result of writing to the GPIODATA register, or it can reflect the previous state of the pin if the pin is switched to GPIO output from GPIO input or another digital function. A read returns the current state of the output latch.
- If a pin is configured as another digital function (input or output), a write to the GPIODATA register has no effect on the pin level. A read returns the current state of the pin even if it is configured as an output. This means that by reading the GPIODATA register, the digital output or input value of a function other than GPIO on that pin can be observed.

The following rules apply when the pins are switched from input to output:

- Pin is configured as input with a HIGH level applied:
  – Change pin to output: pin drives HIGH level.
- Pin is configured as input with a LOW level applied:
  – Change pin to output: pin drives LOW level.

The rules show that the pins mirror the current logic level. Therefore floating pins may drive an unpredictable level when switched from input to output.

9.4.2 GPIO data direction register

Table 150. GPIO data direction register (GPIO0DIR, address 0x5000 8000 to GPIO3DIR, address 0x5003 8000) bit description

Bit	Symbol	Description	Reset value	Access
11:0	IO	Selects pin x as input or output (x = 0 to 11). 0 = Pin PIOn_x is configured as input. 1 = Pin PIOn_x is configured as output.	0x00	R/W
31:12	-	Reserved	-	-

9.4.3 GPIO interrupt sense register

Table 151. GPIO interrupt sense register (GPIO0IS, address 0x5000 8004 to GPIO3IS, address 0x5003 8004) bit description

Bit	Symbol	Description	Reset value	Access
11:0	ISENSE	Selects interrupt on pin x as level or edge sensitive (x = 0 to 11). 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
31:12	-	Reserved	-	-

9.4.4 GPIO interrupt both edges sense register

Table 152. GPIO interrupt both edges sense register (GPIO0IBE, address 0x5000 8008 to GPIO3IBE, address 0x5003 8008) bit description

Bit	Symbol	Description	Reset value	Access
11:0	IBE	Selects interrupt on pin x to be triggered on both edges (x = 0 to 11). 0 = Interrupt on pin PIOn_x is controlled through register GPIOIEV. 1 = Both edges on pin PIOn_x trigger an interrupt.	0x00	R/W
31:12	-	Reserved	-	-

9.4.5 GPIO interrupt event register

Table 153. GPIO interrupt event register (GPIO0IEV, address 0x5000 800C to GPIO3IEV, address 0x5003 800C) bit description

Bit	Symbol	Description	Reset value	Access
11:0	IEV	Selects interrupt on pin x to be triggered rising or falling edges (x = 0 to 11). 0 = Depending on setting in register GPIOIS (see Table 151), falling edges or LOW level on pin PIOn_x trigger an interrupt. 1 = Depending on setting in register GPIOIS (see Table 151), rising edges or HIGH level on pin PIOn_x trigger an interrupt.	0x00	R/W
31:12	-	Reserved	-	-

9.4.6 GPIO interrupt mask register

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

Table 154. GPIO interrupt mask register (GPIO0IE, address 0x5000 8010 to GPIO3IE, address 0x5003 8010) bit description

Bit	Symbol	Description	Reset value	Access
11:0	MASK	Selects interrupt on pin x to be masked (x = 0 to 11). 0 = Interrupt on pin PIOn_x is masked. 1 = Interrupt on pin PIOn_x is not masked.	0x00	R/W
31:12	-	Reserved	-	-

9.4.7 GPIO raw interrupt status register

Bits read HIGH in the GPIORS 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 GPIOIE. Bits read as zero indicate that the corresponding input pins have not initiated an interrupt. The register is read-only.

Table 155. GPIO raw interrupt status register (GPIO0RIS, address 0x5000 8014 to GPIO3RIS, address 0x5003 8014) bit description

Bit	Symbol	Description	Reset value	Access
11:0	RAWST	Raw interrupt status (x = 0 to 11). 0 = No interrupt on pin PIOn_x. 1 = Interrupt requirements met on PIOn_x.	0x00	R
31:12	-	Reserved	-	-

9.4.8 GPIO masked interrupt status register

Bits read HIGH in the GPIOMIS 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. GPIOMIS is the state of the interrupt after masking. The register is read-only.

Table 156. GPIO masked interrupt status register (GPIO0MIS, address 0x5000 8018 to GPIO3MIS, address 0x5003 8018) bit description

Bit	Symbol	Description	Reset value	Access
11:0	MASK	Selects interrupt on pin x to be masked (x = 0 to 11). 0 = No interrupt or interrupt masked on pin PIOn_x. 1 = Interrupt on PIOn_x.	0x00	R
31:12	-	Reserved	-	-

9.4.9 GPIO interrupt clear register

This register allows software to clear edge detection for port bits that are identified as edge-sensitive in the Interrupt Sense register. This register has no effect on port bits identified as level-sensitive.

Table 157. GPIO interrupt clear register (GPIO0IC, address 0x5000 801C to GPIO3IC, address 0x5003 801C) bit description

Bit	Symbol	Description	Reset value	Access
11:0	CLR	Selects interrupt on pin x to be cleared (x = 0 to 11). Clears the interrupt edge detection logic. This register is write-only. **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. 0 = No effect. 1 = Clears edge detection logic for pin PIOn_x.	0x00	W
31:12	-	Reserved	-	-

9.5 Functional description

9.5.1 Write/read data operations

In order for software to be able to set GPIO bits without affecting any other pins in a single write operation, bits [13:2] of a 14-bit wide address bus are used to create a 12-bit wide mask for write and read operations on the 12 GPIO pins for each port. Only GPIODATA bits masked by 1 are affected by read and write operations. The masked GPIODATA register can be located anywhere between address offsets 0x0000 to 0x3FFC in the GPIO address space. Reading and writing to the GPIODATA register at address 0x3FFC sets all masking bits to 1.

Write operation

If the address bit (i+2) associated with the GPIO port bit i (i = 0 to 11) to be written is HIGH, the value of the GPIODATA register bit i is updated. If the address bit (i+2) is LOW, the corresponding GPIODATA register bit i is left unchanged.

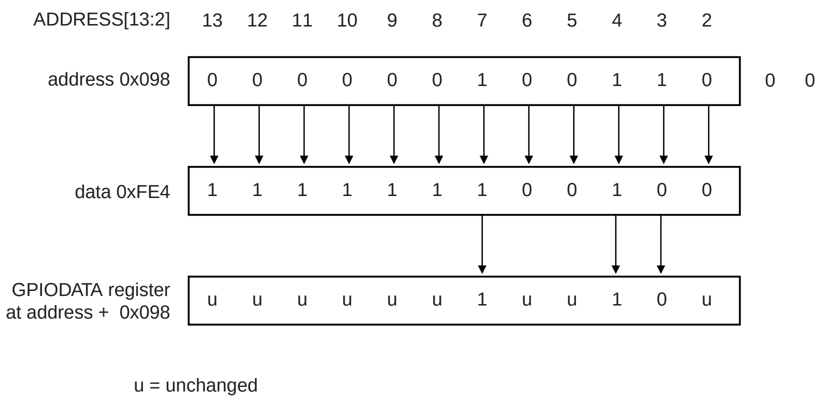

ADDRESS[13:2] 13 12 11 10 9 8 7 6 5 4 3 2
address 0x098 0 0 0 0 0 0 1 0 0 1 1 0 0 0
data 0xFE4 1 1 1 1 1 1 1 0 0 1 0 0
GPIODATA register at address + 0x098 u u u u u u 1 u u 1 0 u
u = unchanged

Fig 14. Masked write operation to the GPIODATA register

Read operation

If the address bit associated with the GPIO data bit is HIGH, the value is read. If the address bit is LOW, the GPIO data bit is read as 0: Reading a port DATA register yields the state of port pins 11:0 ANDed with address bits 13:2.

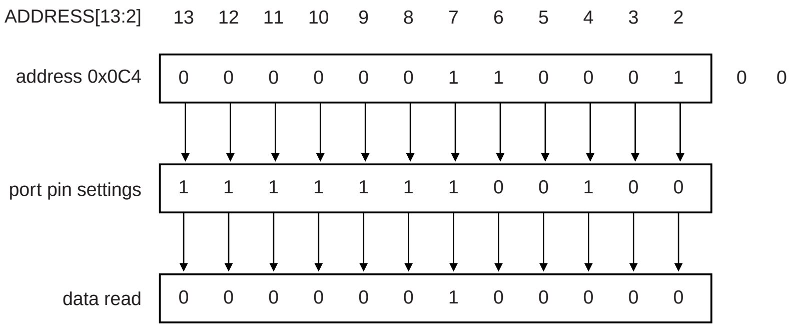

ADDRESS[13:2] 13 12 11 10 9 8 7 6 5 4 3 2
address 0x0C4 0 0 0 0 0 0 1 1 0 0 0 1 0 0
port pin settings 1 1 1 1 1 1 1 0 0 1 0 0
data read 0 0 0 0 0 0 1 0 0 0 0 0

Fig 15. Masked read operation

UM10375

Chapter 10: LPC13xx USB device controller

Rev. 5 — 21 June 2012 **User manual**

10.1 How to read this chapter

The USB device controller is available on parts LPC1342 and LPC1343 only.

10.2 Basic configuration

The USB device is configured using the following registers:

1. Pins: The USB pins must be configured in the IOCONFIG register block (Section 7.4).
2. Power: In the SYSAHBCLKCTRL register, set bit 14 (Table 25) for the USB register interface. The USB PHY must be powered through the PDRUNCFG register (Table 55)
3. Clock: Enable the USB clock by writing to the USBCLKDIV register (Table 33). The USB clock can be selected from the dedicated USB PLL or the main clock (Table 31). For details see Section 10.12.2.

10.3 Introduction

The Universal Serial Bus (USB) is a four-wire bus that supports communication between a host and one or more (up to 127) peripherals. The host controller allocates the USB bandwidth to attached devices through a token-based protocol. The bus supports hot plugging and dynamic configuration of the devices. All transactions are initiated by the host controller.

The host schedules transactions in 1 ms frames. Each frame contains a Start-Of-Frame (SOF) marker and transactions that transfer data to or from device endpoints. Each device can have a maximum of 5 logical or 10 physical endpoints. There are four types of transfers defined for the endpoints. Control transfers are used to configure the device. Interrupt transfers are used for periodic data transfer. Bulk transfers are used when the rate of transfer is not critical. Isochronous transfers have guaranteed delivery time but no error correction.

For more information on the Universal Serial Bus, see the USB Implementers Forum website.

The USB device controller on the LPC134x enables full-speed (12 Mb/s) data exchange with a USB host controller.

Table 158. USB related acronyms, abbreviations, and definitions used in this chapter

Acronym/abbreviation	Description
AHB	Advanced High-performance bus
ATLE	Auto Transfer Length Extraction
ATX	Analog Transceiver
EOP	End-Of-Packet
EP	Endpoint
EP_RAM	Endpoint RAM

Table 158. USB related acronyms, abbreviations, and definitions used in this chapter

Acronym/abbreviation	Description
FS	Full-speed
LED	Light Emitting Diode
LS	Low Speed
MPS	Maximum Packet Size
NAK	Negative Acknowledge
PLL	Phase Locked Loop
RAM	Random Access Memory
SOF	Start-Of-Frame
SIE	Serial Interface Engine
SRAM	Synchronous RAM
UDCA	USB Device Communication Area
USB	Universal Serial Bus

10.4 Features

- Fully compliant with the USB 2.0 specification (full-speed).
- Supports 10 physical (5 logical) endpoints.
- Supports Control, Bulk, Interrupt and Isochronous endpoints.
- Supports SoftConnect feature.
- Double-buffer implementation for one Bulk and one Isochronous endpoint.

10.5 Fixed endpoint configuration

Table 159 shows the supported endpoint configurations. The packet size is fixed for each type of end point.

Table 159. Fixed endpoint configuration

Logical endpoint	Physical endpoint	Endpoint type	Direction	Packet size (byte)	Double-buffer
0	0	Control	Out	64	No
0	1	Control	In	64	No
1	2	Interrupt/Bulk	Out	64	No
1	3	Interrupt/Bulk	In	64	No
2	4	Interrupt/Bulk	Out	64	No
2	5	Interrupt/Bulk	In	64	No
3	6	Interrupt/Bulk	Out	64	Yes
3	7	Interrupt/Bulk	In	64	Yes
4	8	Isochronous	Out	512	Yes
4	9	Isochronous	In	512	Yes

10.6 General description

The architecture of the USB device controller is shown below in Figure 16.

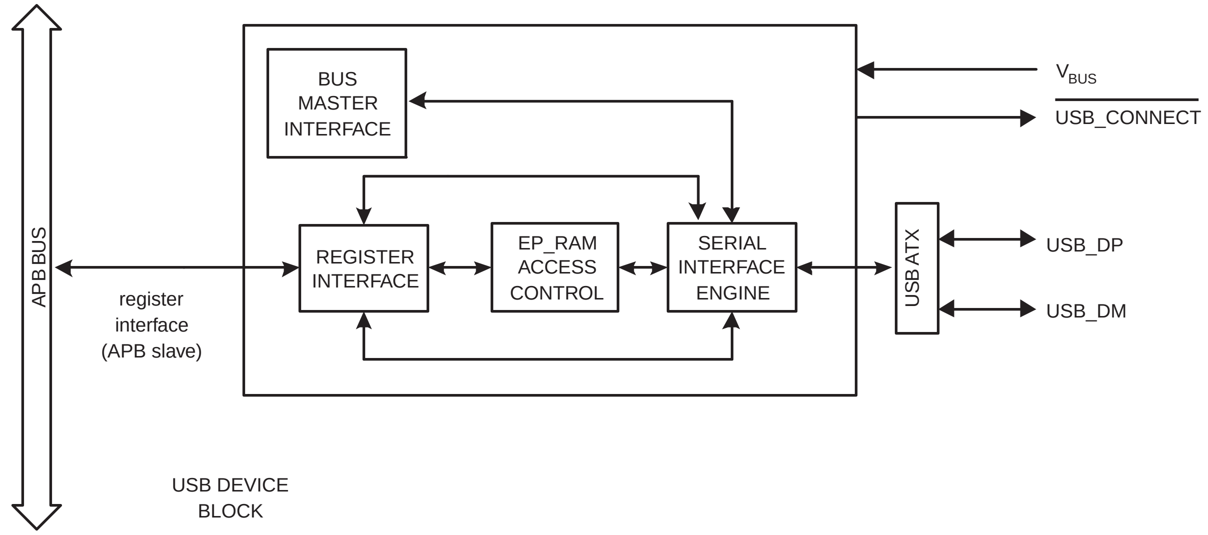


Fig 16. USB device controller block diagram

10.6.1 Analog transceiver

The USB Device Controller has a built-in analog transceiver (ATX). The USB ATX sends/receives the bi-directional USB_DP and USB_DM signals of the USB bus.

10.6.2 Serial Interface Engine (SIE)

The SIE implements the full USB protocol layer. It is completely hardwired for speed and needs no firmware intervention. It handles transfer of data between the endpoint buffers in EP_RAM and the USB bus. The functions of this block include: synchronization pattern recognition, parallel/serial conversion, bit stuffing/de-stuffing, CRC checking/generation, PID verification/generation, address recognition, and handshake evaluation/generation.

10.6.3 Endpoint RAM (EP_RAM)

Each endpoint buffer is implemented as an SRAM based FIFO. The SRAM dedicated for this purpose is called the EP_RAM. Each endpoint has a reserved space in the EP_RAM. The total EP_RAM space is fixed. All endpoints are realized automatically.

10.6.4 EP_RAM access control

The EP_RAM Access Control logic handles transfer of data from/to the EP_RAM and the sources that can access it: the CPU (via the Register Interface) and the SIE.

10.6.5 Register interface

The Register Interface allows the CPU to control the operation of the USB Device Controller. It also provides a way to write transmit data to the controller and read receive data from the controller.

10.6.6 SoftConnect

The connection to the USB is accomplished by bringing USB_DP (for a full-speed device) HIGH through a 1.5 kOhm pull-up resistor. The SoftConnect feature can be used to allow software to finish its initialization sequence before deciding to establish connection to the USB. Re-initialization of the USB bus connection can also be performed without having to unplug the cable.

To use the SoftConnect feature, the $\overline{\text{USB\_CONNECT}}$ signal should control an external switch that connects the 1.5 kOhm resistor between USB_DP and 3.3 V. Software can then control the $\overline{\text{USB\_CONNECT}}$ signal by writing to the CON bit using the SIE Set Device Status command.

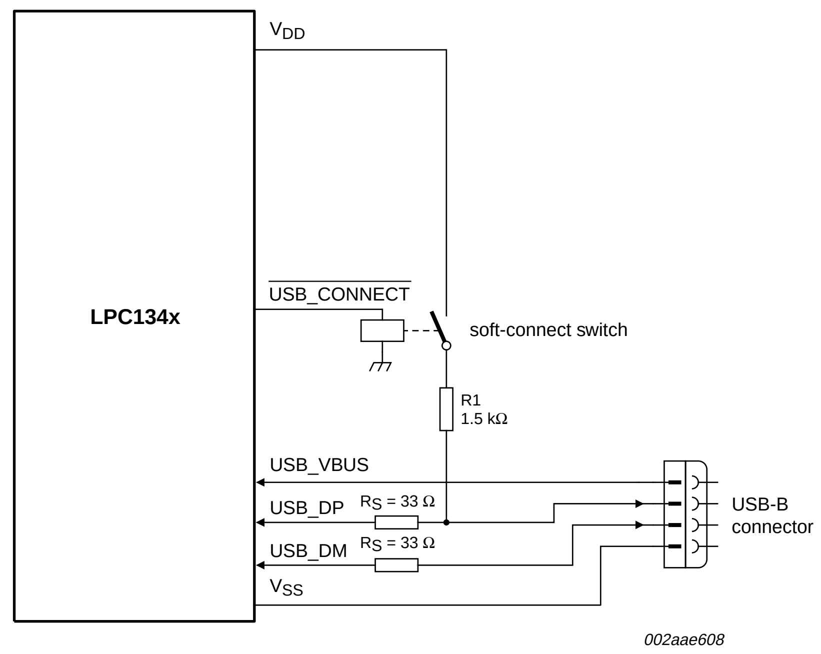


Fig 17. USB SoftConnect interfacing

10.7 Operational overview

Transactions on the USB bus transfer data between device endpoints and the host. The direction of a transaction is defined with respect to the host. OUT transactions transfer data from the host to the device. IN transactions transfer data from the device to the host. All transactions are initiated by the host controller.

For an OUT transaction, the USB ATX receives the bi-directional USB_DP and USB_DM signals of the USB bus. The Serial Interface Engine (SIE) receives the serial data from the ATX and converts it into a parallel data stream. The parallel data is written to the corresponding endpoint buffer.

For IN transactions, the SIE reads the parallel data from the endpoint buffer in EP_RAM, converts it into serial data, and transmits it onto the USB bus using the USB ATX.

Once data has been received or sent, the endpoint buffer can be read or written. The CPU transfers data between RAM and the endpoint buffer using the register interface. See Section 10.13 “Functional description” for a detailed description.

10.8 Pin description

The device controller can access one USB port.

Table 160. USB device pin description

Name	Direction	Description
V_{BUS}	I	V_{BUS} status input. When this function is not enabled via its corresponding IOCONFIG register, it is driven HIGH internally.
$\overline{\text{USB\_CONNECT}}$	O	SoftConnect control signal.
USB_FTOGGLE	O	USB 1 ms SoF signal.
USB_DP	I/O	Positive differential data.
USB_DM	I/O	Negative differential data.

10.9 Clocking and power control

This section describes the clocking and power management features of the USB Device Controller.

10.9.1 Power requirements

The USB protocol insists on power management by the device. This becomes very critical if the device draws power from the bus (bus-powered device). The following constraints should be met by a bus-powered device:

1. A device in the non-configured state should draw a maximum of 100 mA from the bus.
2. A configured device can draw only up to what is specified in the Max Power field of the configuration descriptor. The maximum value is 500 mA.
3. A suspended device can draw a maximum of 500 μA.

10.9.2 Clocks

The USB device controller clocks are shown in Table 161

Table 161. USB device controller clock sources

Source	Clock name	Description
ahb_sys_clk	PCLK	This is the system clock. Minimum frequency of this clock is 16 MHz.
usb_clk (see Table 12)	USB_MainClk	USB_MainClk is the 48 MHz ± 500 ppm input clock. This clock does not need to be synchronized with the system clock (PCLK). Gating of this clock is possible by an external control block using the USB_NeedClk signal. This clock will be used to recover the 12 MHz clock from the USB bus

The usb_clk clock can be either provided by the main clock or a dedicated USB PLL (see Figure 3). The USB PLL can be powered down if it is not used for the usb_clk in the PDRUNCFG register (Table 55) to conserve power.

10.9.3 Power management support

To help conserve power, the USB device controller automatically disables PCLK and USB_MainClk when not in use.

The assertion of USB_Suspend(_N) signal indicates that there was no activity on the USB bus for the last 3 ms. At this time an interrupt is sent to the processor on which the software can start preparing the device for suspend.

If there is no activity again for the next 2 ms, the USB_NeedClk signal will go low. This shuts off the USB_MainClk automatically. Once the USB_MainClk is switched off, internal registers in the USB clock domain will not be visible to the software.

When the activity is detected on the bus, USB_Suspend(_N) is deactivated and USB_NeedClk signal is activated. This process is fully combinatorial and hence no USB_MainClk is required to activate the USB_NeedClk signal.

The usb_clk_enable signal is provided by the SYSAHBCLK bit 14 (see Table 25) which enables the clock to the USB register block.

In addition, the on-chip device PHY can be powered down in the PDRUNCFG register (Table 55) if the USB device function is not needed.

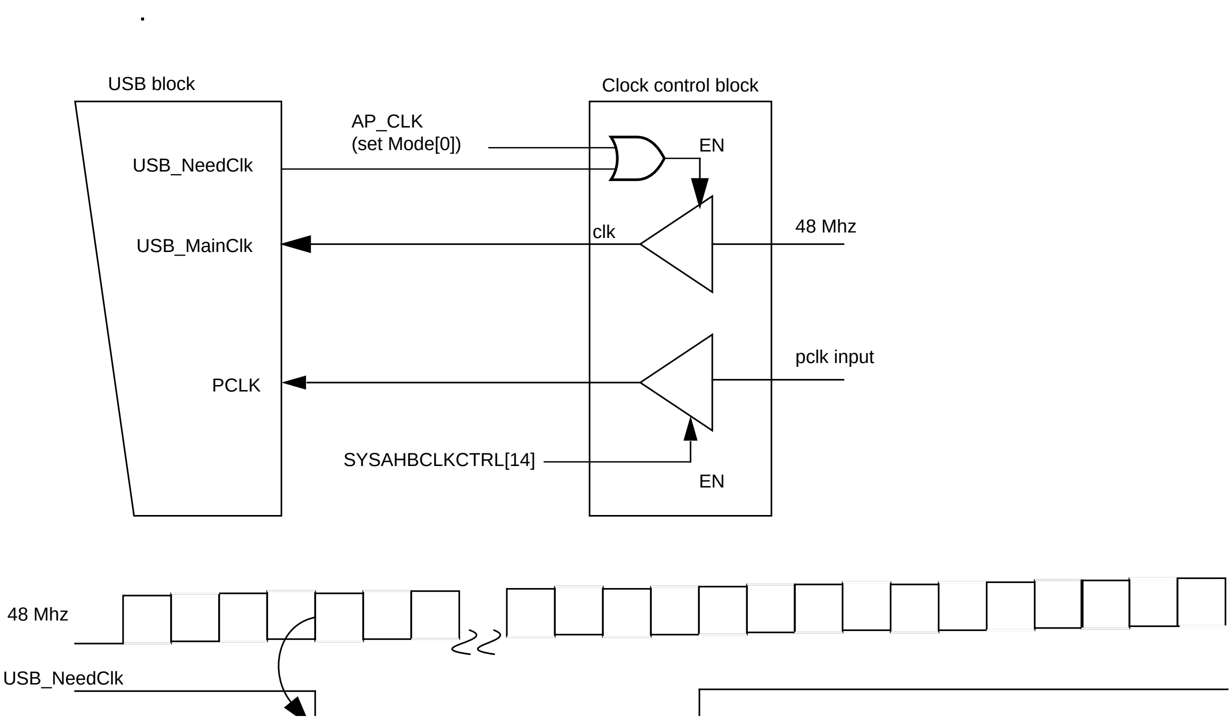


Fig 18. USB clocking

10.9.4 Remote wake-up

The USB block supports software initiated remote wake-up. Remote wake-up involves a resume signal initiated from the device. This is done by resetting the suspend bit in the Device Status register. Before writing into the register, both the USB_MainClk and PCLK need to be enabled in the system control block.

Before the device is suspended, it is important that the AP_CLK bit in the Set Mode register is set. The USB PHY should not be disabled while the device is suspended so it can continue to respond to USB bus events.

10.9.5 Interrupts

The external interrupt generation takes place only if the necessary ‘enable’ bits are set in the Device Interrupt Enable register. The raw interrupt status will be registered in the status register. The interrupt has to be cleared by writing ‘1’ into the interrupt clear register.

10.10 Register description

Table 162 shows the USB Device Controller registers directly accessible by the CPU. The Serial Interface Engine (SIE) has other registers that are indirectly accessible via the SIE command registers. See Section 10.11 “Serial interface engine command description” for more information.

Table 162. Register overview: USB device (base address 0x4002 0000)

Name	Access	Address offset	Description	Reset value
Device interrupt registers				
USBDevIntSt	RO	0x00	USB Device Interrupt Status	0x0000 0010
USBDevIntEn	R/W	0x04	USB Device Interrupt Enable	0x0000 0000
USBDevIntClr	WO	0x08	USB Device Interrupt Clear	0x0000 0000
USBDevIntSet	WO	0x0C	USB Device Interrupt Set	0x0000 0000
SIE command registers				
USBCmdCode	WO	0x10	USB Command Code	0x0000 0000
USBCmdData	RO	0x14	USB Command Data	0x0000 0000
USB data transfer registers				
USBRxData	RO	0x18	USB Receive Data	0x0000 0000
USBTxData	WO	0x1C	USB Transmit Data	0x0000 0000
USBRxPLen	RO	0x20	USB Receive Packet Length	0x0000 0000
USBTxPLen	WO	0x24	USB Transmit Packet Length	0x0000 0000
USBCtrl	R/W	0x28	USB Control	0x0000 0000
Miscellaneous registers				
USBDevFIQSel	WO	0x2C	USB Device FIQ select	0x00

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

[2] Reading WO register will return an invalid value.

10.10.1 Device interrupt registers

10.10.1.1 USB Device Interrupt Status register (USBDevIntSt - 0x4002 0000)

The USBDevIntSt register holds the status of each interrupt whether it is enabled or not. A 0 indicates no interrupt and 1 indicates the presence of the interrupt. USBDevIntSt is a read only register.

Table 163. USB Device Interrupt Status register (USBDevIntSt - address 0x4002 0000) bit description

Bit	Symbol	Description	Reset value
0	FRAME	The frame interrupt occurs every 1 ms. This is used in isochronous packet transfers. 0 = no interrupt. 1 = interrupt pending.	-
1	EP0	USB core interrupt for physical endpoint 0. 0 = no interrupt. 1 = interrupt pending.	-
2	EP1	USB core interrupt for physical endpoint 1. 0 = no interrupt. 1 = interrupt pending.	-
3	EP2	USB core interrupt for physical endpoint 2. 0 = no interrupt. 1 = interrupt pending.	-

Table 163. USB Device Interrupt Status register (USBDevIntSt - address 0x4002 0000) bit description *…continued*

Bit	Symbol	Description	Reset value
4	EP3	USB core interrupt for physical endpoint 3. 0 = no interrupt. 1 = interrupt pending.	-
5	EP4	USB core interrupt for physical endpoint 4. 0 = no interrupt. 1 = interrupt pending.	-
6	EP5	USB core interrupt for physical endpoint 5. 0 = no interrupt. 1 = interrupt pending.	-
7	EP6	USB core interrupt for physical endpoint 6. 0 = no interrupt. 1 = interrupt pending.	-
8	EP7	USB core interrupt for physical endpoint 7. 0 = no interrupt. 1 = interrupt pending.	-
9	DEV_STAT	Set when USB Bus reset, USB suspend change, or Connect change event occurs. Refer to Section 10.11.7. 0 = no interrupt. 1 = interrupt pending.	0
10	CC_EMPTY	The command code register (USBCmdCode) is empty (New command can be written). 0 = no interrupt. 1 = interrupt pending.	1
11	CD_FULL	Command data register (USBCmdData) is full (Data can be read now). 0 = no interrupt. 1 = interrupt pending.	0
12	RxENDPKT	The current packet in the endpoint buffer is transferred to the CPU. 0 = no interrupt. 1 = interrupt pending.	0
13	TxENDPKT	The number of data bytes transferred to the endpoint buffer equals the number of bytes programmed in the TxPacket length register (USBTxPLen). 0 = no interrupt. 1 = interrupt pending.	0
31:14	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

10.10.1.2 USB Device Interrupt Enable register (USBDevIntEn - 0x4002 0004)

Writing a one to a bit in this register enables the corresponding bit in USBDevIntSt to generate an external interrupt when set. If it’s not set, no external interrupt is generated, but the interrupt will still be held in the Device Interrupt Status register.

Table 164. USB Device Interrupt Enable register (USBDevIntEn - address 0x4002 0004) bit description

Bit	Symbol	Description	Reset value
0	FRAME_EN	Frame interrupt. For isochronous packet transfers. 0 = no interrupt generated. 1 = interrupt generated when the corresponding bit in USBDevIntSt is set.	0
1	EP0_EN	USB core interrupt for physical endpoint 0. 0 = no interrupt generated. 1 = interrupt generated when the corresponding bit in USBDevIntSt is set.	0
2	EP1_EN	USB core interrupt for physical endpoint 1. 0 = no interrupt generated. 1 = interrupt generated when the corresponding bit in USBDevIntSt is set.	0
3	EP2_EN	USB core interrupt for physical endpoint 2. 0 = no interrupt generated. 1 = interrupt generated when the corresponding bit in USBDevIntSt is set.	0
4	EP3_EN	USB core interrupt for physical endpoint 3. 0 = no interrupt generated. 1 = interrupt generated when the corresponding bit in USBDevIntSt is set.	0
5	EP4_EN	USB core interrupt for physical endpoint 4. 0 = no interrupt generated. 1 = interrupt generated when the corresponding bit in USBDevIntSt is set.	0
6	EP5_EN	USB core interrupt for physical endpoint 5. 0 = no interrupt. 1 = interrupt pending.	0
7	EP6_EN	USB core interrupt for physical endpoint 6. 0 = no interrupt generated. 1 = interrupt generated when the corresponding bit in USBDevIntSt is set.	0
8	EP7_EN	USB core interrupt for physical endpoint 7. 0 = no interrupt generated. 1 = interrupt generated when the corresponding bit in USBDevIntSt is set.	0
9	DEV_STAT_EN	Set when USB Bus reset, USB suspend change, or Connect change event occurs. 0 = no interrupt generated. 1 = interrupt generated when the corresponding bit in USBDevIntSt is set.	0
10	CC_EMPTY_EN	The command code register (USBCmdCode) is empty (New command can be written). 0 = no interrupt generated. 1 = interrupt generated when the corresponding bit in USBDevIntSt is set.	0

Table 164. USB Device Interrupt Enable register (USBDevIntEn - address 0x4002 0004) bit description

Bit	Symbol	Description	Reset value
11	CD_FULL_EN	Command data register (USBCmdData) is full (Data can be read now). 0 = no interrupt generated. 1 = interrupt generated when the corresponding bit in USBDevIntSt is set.	0
12	RXENDPKT_EN	The current packet in the endpoint buffer is transferred to the CPU. 0 = no interrupt generated. 1 = interrupt generated when the corresponding bit in USBDevIntSt is set.	0
13	TXENDPKT_EN	The number of data bytes transferred to the endpoint buffer equals the number of bytes programmed in the TxPacket length register (USBTxPLen). 0 = no interrupt generated. 1 = interrupt generated when the corresponding bit in USBDevIntSt is set.	0
31:14	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

10.10.1.3 USB Device Interrupt Clear register (USBDevIntClr - 0x4002 0008)

Writing one to a bit in this register clears the corresponding bit in USBDevIntSt. Writing a zero has no effect.

USBDevIntClr is a write only register.

Table 165. USB Device Interrupt Clear register (USBDevIntClr - address 0x4002 0008) bit description

Bit	Symbol	Description	Reset value
0	FRAME_CLR	Frame interrupt. For isochronous packet transfers. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is cleared.	0
1	EP0_CLR	USB core interrupt for physical endpoint 0. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is cleared.	0
2	EP1_CLR	USB core interrupt for physical endpoint 1. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is cleared.	0
3	EP2_CLR	USB core interrupt for physical endpoint 2. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is cleared.	0
4	EP3_CLR	USB core interrupt for physical endpoint 3. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is cleared.	0
5	EP4_CLR	USB core interrupt for physical endpoint 4. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is cleared.	0

Table 165. USB Device Interrupt Clear register (USBDevIntClr - address 0x4002 0008) bit description

Bit	Symbol	Description	Reset value
6	EP5_CLR	USB core interrupt for physical endpoint 5. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is cleared.	0
7	EP6_CLR	USB core interrupt for physical endpoint 6. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is cleared.	0
8	EP7_CLR	USB core interrupt for physical endpoint 7. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is cleared.	0
9	DEV_STAT_CLR	Set when USB Bus reset, USB suspend change, or Connect change event occurs. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is cleared.	0
10	CC_EMPTY_CLR	The command code register (USBCmdCode) is empty (New command can be written). 0 = no effect. 1 = the corresponding bit in USBDevIntSt is cleared.	0
11	CD_FULL_CLR	Command data register (USBCmdData) is full (Data can be read now). 0 = no effect. 1 = the corresponding bit in USBDevIntSt is cleared.	0
12	RXENDPKT_CLR	The current packet in the endpoint buffer is transferred to the CPU. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is cleared.	0
13	TXENDPKT_CLR	The number of data bytes transferred to the endpoint buffer equals the number of bytes programmed in the TxPacket length register (USBTxPLen). 0 = no effect. 1 = the corresponding bit in USBDevIntSt is cleared.	0
31:14	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

10.10.1.4 USB Device Interrupt Set register (USBDevIntSet - 0x4002 000C)

Writing one to a bit in this register sets the corresponding bit in the USBDevIntSt. Writing a zero has no effect

USBDevIntSet is a write only register.

Table 166. USB Device Interrupt Set register (USBDevIntSet - address 0x4002 000C) bit description

Bit	Symbol	Description	Reset value
0	FRAME_SET	Frame interrupt. For isochronous packet transfers. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is set.	0
1	EP0_SET	USB core interrupt for physical endpoint 0. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is set.	0
2	EP1_SET	USB core interrupt for physical endpoint 1. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is set.	0
3	EP2_SET	USB core interrupt for physical endpoint 2. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is set.	0
4	EP3_SET	USB core interrupt for physical endpoint 3. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is set.	0
5	EP4_SET	USB core interrupt for physical endpoint 4. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is set.	0
6	EP5_SET	USB core interrupt for physical endpoint 5. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is set.	0
7	EP6_SET	USB core interrupt for physical endpoint 6. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is set.	0
8	EP7_SET	USB core interrupt for physical endpoint 7. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is set.	0
9	DEV_STAT_SET	Set when USB Bus reset, USB suspend change, or Connect change event occurs. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is set.	0
10	CC_EMPTY_SET	The command code register (USBCmdCode) is empty (New command can be written). 0 = no effect. 1 = the corresponding bit in USBDevIntSt is set.	0
11	CD_FULL_SET	Command data register (USBCmdData) is full (Data can be read now). 0 = no effect. 1 = the corresponding bit in USBDevIntSt is set.	0

Table 166. USB Device Interrupt Set register (USBDevIntSet - address 0x4002 000C) bit description

Bit	Symbol	Description	Reset value
12	RXENDPKT_SET	The current packet in the endpoint buffer is transferred to the CPU. 0 = no effect. 1 = the corresponding bit in USBDevIntSt is set.	0
13	TXENDPKT_SET	The number of data bytes transferred to the endpoint buffer equals the number of bytes programmed in the TxPacket length register (USBTxPLen). 0 = no effect. 1 = the corresponding bit in USBDevIntSt is set.	0
31:14	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

10.10.2 SIE command code registers

The SIE command code registers are used for communicating with the Serial Interface Engine. See Section 10.11 “Serial interface engine command description” for more information.

10.10.2.1 USB Command Code register (USBCmdCode - 0x4001 8010)

This register is used for sending the command and write data to the SIE. The commands written here are propagated to the SIE and executed there. After executing the command, the register is empty, and the CCEMPTY bit of USBDevIntSt register is set. See Section 10.11 for details. USBCmdCode is a write only register.

Table 167. USB Command Code register (USBCmdCode - address 0x4002 0010) bit description

Bit	Symbol	Value	Description	Reset value
7:0	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
15:8	CMD_PHASE		Command phase action	0x00
		0x01	Write	
		0x02	Read	
		0x05	Command	
23:16	CODE_WDATA		This is a multi-purpose field. When CMD_PHASE is Command or Read, this field contains the code for the command (CMD_CODE). When CMD_PHASE is Write, this field contains the command write data (CMD_WDATA).	0x00
31:24	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

10.10.2.2 USB Command Data register (USBCmdData - 0x4002 0014)

This register contains the data retrieved after executing a SIE command. When the data is ready to be read, the CD_FULL bit of the USBDevIntSt register is set. USBCmdData is a read only register.

Table 168. USB Command Data register (USBCmdData - address 0x4002 0014) bit description

Bit	Symbol	Description	Reset value
7:0	CMD_RDATA	Command Read Data.	0x00
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.10.3 USB data transfer registers

The registers in this group are used for transferring data between endpoint buffers and RAM in Slave mode operation. See Section 10.13 “Functional description”.

10.10.3.1 USB Receive Data register (USBRxData - 0x4002 0018)

For an OUT transaction, the CPU reads the endpoint buffer data from this register. Before reading this register, the RD_EN bit and LOG_ENDPOINT field of the USBCtrl register should be set appropriately. On reading this register, data from the selected endpoint buffer is fetched. The data is in little endian format: the first byte received from the USB bus will be available in the least significant byte of USBRxData. USBRxData is a read only register.

Table 169. USB Receive Data register (USBRxData - address 0x4002 0018) bit description

Bit	Symbol	Description	Reset value
31:0	RX_DATA	Data received.	0x0000 0000

10.10.3.2 USB Transmit Data register (USBTxData - 0x4002 021C)

For an IN transaction, the CPU writes the endpoint data into this register. Before writing to this register, the WR_EN bit and LOG_ENDPOINT field of the USBCtrl register should be set appropriately, and the packet length should be written to the USBTxPlen register. On writing this register, the data is written to the selected endpoint buffer. The data is in little endian format: the first byte sent on the USB bus will be the least significant byte of USBTxData. USBTxData is a write only register.

Table 170. USB Transmit Data register (USBTxData - address 0x4002 001C) bit description

Bit	Symbol	Description	Reset value
31:0	TX_DATA	Transmit Data.	0x0000 0000

10.10.3.3 USB Receive Packet Length register (USBRxPLen - 0x4002 0020)

This gives the number of bytes remaining in the RAM for the current packet being transferred and whether the packet is valid or not. This register will get updated at every word that gets transferred to the system. The processor can use this register to get the number of bytes to be transferred. When the number of bytes reaches zero, an end of packet interrupt is generated.

Table 171. USB Receive Packet Length register (USBRxPLen - address 0x4002 0020) bit description

Bit	Symbol	Value	Description	Reset value
9:0	PKT_LNGTH	-	The remaining number of bytes to be read from the currently selected endpoint's buffer. When this field decrements to 0, the RxENDPKT bit will be set in USBDevIntSt.	0
10	DV		Data valid. This bit is useful for isochronous endpoints. Non-isochronous endpoints do not raise an interrupt when an erroneous data packet is received. But invalid data packet can be produced with a bus reset. For isochronous endpoints, data transfer will happen even if an erroneous packet is received. In this case DV bit will not be set for the packet.	0
		0	Data is invalid.	
		1	Data is valid.	
31:11	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.10.3.4 USB Transmit Packet Length register (USBTxPLen - 0x4002 0024)

This indicates the number of bytes still to be transferred from the processor to the RAM. The processor has to program this register with the byte length of the packet to be sent, before writing to the Transmit Data Register. The processor can read this register to determine the number of bytes it has transferred to the memory.

Remark: To transfer an empty packet, this register has to be set to 0x00 and a single write operation has to be performed on the Transmit Data Register.

Table 172. USB Transmit Packet Length register (USBTxPLen - address 0x4002 0024) bit description

Bit	Symbol	Description	Reset value
9:0	PKT_LNGTH	The remaining number of bytes to be written to the selected endpoint buffer. This field is decremented by 4 by hardware after each write to USBTxData. When this field decrements to 0, the TxENDPKT bit will be set in USBDevIntSt.	0
31:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

10.10.3.5 USB Control register (USBCtrl - 0x4002 0028)

This register controls the data transfer operation of the USB device. It selects the endpoint buffer that is accessed by the USBRxData and USBTxData registers and enables reading and writing them. USBCtrl is a read/write register.

Table 173. USB Control register (USBCtrl - address 0x4002 0028) bit description

Bit	Symbol	Value	Description	Reset value
0	RD_EN		Read mode control. Enables reading data from the OUT endpoint buffer for the endpoint specified in the LOG_ENDPOINT field using the USBRxData register. This bit is cleared by hardware when the last word of the current packet is read from USBRxData.	0
		0	Read mode is disabled.	
		1	Read mode is enabled.	
1	WR_EN		Write mode control. Enables writing data to the IN endpoint buffer for the endpoint specified in the LOG_ENDPOINT field using the USBTxData register. This bit is cleared by hardware when the number of bytes in USBTxLen have been sent.	0
		0	Write mode is disabled.	
		1	Write mode is enabled.	
5:2	LOG_ENDPOINT	-	Logical Endpoint number.	0x0
31:6	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.10.3.5.1 Data transfer

When the software wants to read the data from an endpoint buffer it should make the Read Enable bit high and should program the logical endpoint number. The control logic will first fetch the packet length to the receive packet length register. Also the hardware fills the receive data register with the first word of the packet.

The software can now start reading the receive data register. When the end of packet is reached the Read Enable bit will be disabled by the control logic and RxENDPKT bit is set in the Device interrupt status register.

If the software makes the Read Enable bit low midway, the reading will be terminated. In this case the data will remain in the RAM. When the Read Enable signal is made high again for this endpoint, data will be read from the beginning.

For writing data to an endpoint buffer, the Write Enable bit should be made high and software should write to the Tx Packet Length register the number of bytes it is going to send in the packet. It can then write data continuously in the Transmit Data register. When the control logic receives the number of bytes programmed in the Tx Packet length register, it will reset the Write Enable bit. If the software resets this bit midway, writing will start again from the beginning.

Both Read Enable and Write Enable bits can be high at the same time for the same logical endpoint. The interleaved read and write operation is possible.

Remark: It takes 3 clock cycle to fetch the packet length from the RAM after programming the USB control register. There can be a corruption on the packet length value read if the reading of the packet length occurs immediately (in the very next clock cycle) after the programming of USB control register. To avoid this problem, a NOP instruction has to be inserted in between the programming of USBCtrl registers and reading of packet length registers.

For example, follow these steps:

1. USBCtrl = 0x01
2. delay(0) -- generate 1 clock cycle delay
3. pkt_length = USBTxPLen or USBRxPlen

10.10.4 Miscellaneous registers

10.10.4.1 USB Device FIQ Select register (USBDevFIQSel - 0x4002 002C)

When a bit is set ‘1’, the corresponding interrupt will be routed to the high priority interrupt line. Setting all bits to ‘1’ at the same time is not allowed. If the software attempts to set all the bits to ‘1’, none of them will be routed to the high priority interrupt line.

Table 174. USB Device FIQ Select register (USBDevFIQSel - address 0x4002 002C) bit description

Bit	Symbol	Value	Description	Reset value
0	FRAME		This interrupt comes from a 1 KHz free running clock resynchronized on the incoming SoF tokens. This is to be used for isochronous packet transfer.	0
		0	FRAME interrupt will be routed to the low-priority interrupt line IRQ.	
		1	FRAME interrupt will be routed to the high-priority interrupt line FIQ.	
1	BULKOUT		Interrupt routing for bulk out endpoints **Remark:** For logical endpoint 3 (physical endpoints 6 and 7) only.	0
		0	BULKOUT interrupt will be routed to the low-priority interrupt line IRQ.	
		1	BULKOUT interrupt will be routed to the high-priority interrupt line FIQ.	
2	BULKIN		Interrupt routing for bulk in endpoints **Remark:** For logical endpoint 3 (physical endpoints 6 and 7) only.	0
		0	BULKIN interrupt will be routed to the low-priority interrupt line IRQ.	
		1	BULKIN interrupt will be routed to the high-priority interrupt line FIQ.	
31:3	-	-	Reserved	-

10.11 Serial interface engine command description

The functions and registers of the Serial Interface Engine (SIE) are accessed using commands, which consist of a command code followed by optional data bytes (read or write action). The USBCmdCode (Table 167) and USBCmdData (Table 168) registers are used for these accesses.

A complete access consists of two phases:

1. **Command phase:** the USBCmdCode register is written with the CMD_PHASE field set to the value 0x05 (Command), and the CMD_CODE field set to the desired command code. On completion of the command, the CCEMPTY bit of USBDevIntSt is set.
2. **Data phase (optional):** for writes, the USBCmdCode register is written with the CMD_PHASE field set to the value 0x01 (Write), and the CMD_WDATA field set with the desired write data. On completion of the write, the CCEMPTY bit of USBDevIntSt is set. For reads, USBCmdCode register is written with the CMD_PHASE field set to the value 0x02 (Read), and the CMD_CODE field set with command code the read corresponds to. On completion of the read, the CDFULL bit of USBDevInSt will be set, indicating the data is available for reading in the USBCmdData register. In the case of multi-byte registers, the least significant byte is accessed first.

An overview of the available commands is given in Table 175.

Here is an example of the Read Current Frame Number command (reading 2 bytes):

```
USBDevIntClr = 0x30;           // Clear both CCEMPTY & CDFULL
USBCmdCode = 0x00F50500;       // CMD_CODE=0xF5, CMD_PHASE=0x05(Command)
while (!(USBDevIntSt & 0x10)); // Wait for CCEMPTY.
USBDevIntClr = 0x10;           // Clear CCEMPTY interrupt bit.
USBCmdCode = 0x00F50200;       // CMD_CODE=0xF5, CMD_PHASE=0x02(Read)
while (!(USBDevIntSt & 0x20)); // Wait for CDFULL.
USBDevIntClr = 0x20;           // Clear CDFULL.
CurFrameNum = USBCmdData;      // Read Frame number LSB byte.
USBCmdCode = 0x00F50200;       // CMD_CODE=0xF5, CMD_PHASE=0x02(Read)
while (!(USBDevIntSt & 0x20)); // Wait for CDFULL.
Temp = USBCmdData;             // Read Frame number MSB byte
USBDevIntClr = 0x20;           // Clear CDFULL interrupt bit.
CurFrameNum = CurFrameNum | (Temp << 8);
```

Here is an example of the Set Address command (writing 1 byte):

```
USBDevIntClr = 0x10;           // Clear CCEMPTY.
USBCmdCode = 0x00D00500;       // CMD_CODE=0xD0, CMD_PHASE=0x05(Command)
while (!(USBDevIntSt & 0x10)); // Wait for CCEMPTY.
USBDevIntClr = 0x10;           // Clear CCEMPTY.
USBCmdCode = 0x008A0100;       // CMD_WDATA=0x8A(DEV_EN=1, DEV_ADDR=0xA),
                               // CMD_PHASE=0x01(Write)
while (!(USBDevIntSt & 0x10)); // Wait for CCEMPTY.
USBDevIntClr = 0x10;           // Clear CCEMPTY.
```

Table 175. SIE command code table

Command name	Recipient	Code (Hex)	Data phase
Device commands			
Set Address	Device	D0	Write 1 byte
Configure Device	Device	D8	Write 1 byte
Set Mode	Device	F3	Write 1 byte
Read Interrupt Status	Device	F4	Read 1 or 2 bytes
Read Current Frame Number	Device	F5	Read 1 or 2 bytes
Read Chip ID	Device	FD	Read 2 bytes

Table 175. SIE command code table

Command name	Recipient	Code (Hex)	Data phase
Set Device Status	Device	FE	Write 1 byte
Get Device Status	Device	FE	Read 1 byte
Get Error Code	Device	FF	Read 1 byte
Endpoint Commands			
Select Endpoint	Endpoint 0	00	Read 1 byte (optional)
	Endpoint 1	01	Read 1 byte (optional)
	Endpoint xx	xx	Read 1 byte (optional)
Select Endpoint/Clear Interrupt	Endpoint 0	40	Read 1 byte
	Endpoint 1	41	Read 1 byte
	Endpoint xx	xx + 40	Read 1 byte
Set Endpoint Status	Endpoint 0	40	Write 1 byte
	Endpoint 1	41	Write 1 byte
	Endpoint xx	xx + 40	Write 1 byte
Clear Buffer	Selected Endpoint	F2	Read 1 byte (optional)
Validate Buffer	Selected Endpoint	FA	None

10.11.1 Set Address (Command: 0xD0, Data: write 1 byte)

The Set Address command is used to set the USB assigned address and enable the (embedded) function. The address set in the device will take effect after the status stage of the control transaction. After a bus reset, DEV_ADDR is set to 0x00, and DEV_EN is set to 1. The device will respond to packets for function address 0x00, endpoint 0 (default endpoint).

Table 176. Device Set Address command description

Bit	Symbol	Description	Reset value
6:0	DEV_ADDR	Device address set by the software. After a bus reset this field is set to 0x00.	0x00
7	DEV_EN	Device Enable. After a bus reset this bit is set to 1. 0: Device will not respond to any packets. 1: Device will respond to packets for function address DEV_ADDR.	0

10.11.2 Configure Device (Command: 0xD8, Data: write 1 byte)

A value of 1 written to the register indicates that the device is configured and all the enabled non-control endpoints will respond. Control endpoints are always enabled and respond even if the device is not configured, in the default state.

Table 177. Configure Device command description

Bit	Symbol	Description	Reset value
0	CONF_DEVICE	Device is configured. All enabled non-control endpoints will respond. This bit is cleared by hardware when a bus reset occurs. When set, the UP_LED signal is driven LOW if the device is not in the suspended state (SUS=0).	
7:1	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.11.3 Set Mode (Command: 0xF3, Data: write 1 byte)

Table 178. Set Mode command description

Bit	Symbol	Value	Description	Reset value
0	AP_CLK		Always PLL Clock.	0
		0	USB_NEED_CLK is functional; the 48 MHz clock can be stopped when the device enters suspend state.	
		1	USB_NEED_CLK is fixed to 1; the 48 MHz clock cannot be stopped when the device enters suspend state.	
1	INAK_CI		Interrupt on NAK for Control IN endpoint.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed IN transactions generate interrupts.	
2	INAK_CO		Interrupt on NAK for Control OUT endpoint.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed OUT transactions generate interrupts.	
3	INAK_AI		Interrupt on NAK for Interrupt or bulk IN endpoint.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed IN transactions generate interrupts.	
4	INAK_AO		Interrupt on NAK for Interrupt or bulk OUT endpoints.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed OUT transactions generate interrupts.	
7:5	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.11.4 Read Interrupt Status (Command: 0xF4, Data: read 2 bytes)

Table 179. Read interrupt Status byte 1 command description

Bit	Symbol	Description	Reset value
0	EP0	EP0 interrupt	0
1	EP1	EP1 interrupt	0
2	EP2	EP2 interrupt	0
3	EP3	EP3 interrupt	0
4	EP4	EP4 interrupt	0
7:5	-	reserved	-

Table 180. Read interrupt Status byte 2 command description

Bit	Symbol	Value	Description	Reset value
1:0	-		reserved	-
2	D_ST		Device Status change interrupt	0
7:3	-		reserved	-

The device status change is cleared by issuing the Get device status command. All other endpoint interrupts are cleared by issuing select endpoint/clear interrupt command.

10.11.5 Read Current Frame Number (Command: 0xF5, Data: read 1 or 2 bytes)

Returns the frame number of the last successfully received SOF. The frame number is eleven bits wide. The frame number returns least significant byte first. In case the user is only interested in the lower 8 bits of the frame number, only the first byte needs to be read.

- In case no SOF was received by the device at the beginning of a frame, the frame number returned is that of the last successfully received SOF.
- In case the SOF frame number contained a CRC error, the frame number returned will be the corrupted frame number as received by the device.

10.11.6 Read Chip ID (Command: 0xFD, Data: read 2 bytes)

The Chip ID is 16-bit wide. It returns the value the chip ID (LSB first).

10.11.7 Set Device Status (Command: 0xFE, Data: write 1 byte)

The Set Device Status command sets bits in the Device Status Register.

Table 181. Set Device Status command description

Bit	Symbol	Value	Description	Reset value
0	CON		The Connect bit indicates the current connect status of the device. It controls the CONNECT output pin, used for SoftConnect. Reading the connect bit returns the current connect status. This bit is cleared by hardware when the V_{BUS} status input is LOW for more than 3 ms. The 3 ms delay filters out temporary dips in the V_{BUS} voltage.	0
		0	Writing a 0 will make the CONNECT pin go HIGH.	
		1	Writing a 1 will make the CONNECT pin go LOW.	
1	CON_CH		Connect Change.	0
		0	This bit is cleared when read.	
		1	This bit is set when the device's pull-up resistor is disconnected because V_{BUS} disappeared. The DEV_STAT interrupt is generated when this bit is 1.	

Table 181. Set Device Status command description

Bit	Symbol	Value	Description	Reset value
2	SUS		Suspend: The Suspend bit represents the current suspend state. When the device is suspended (SUS = 1) and the CPU writes a 0 into it, the device will generate a remote wake-up. This will only happen when the device is connected (CON = 1). When the device is not connected or not suspended, writing a 0 has no effect. Writing a 1 to this bit has no effect.	0
		0	This bit is reset to 0 on any activity.	
		1	This bit is set to 1 when the device hasn’t seen any activity on its upstream port for more than 3 ms.	
3	SUS_CH		Suspend (SUS) bit change indicator. The SUS bit can toggle because: • The device goes into the suspended state. • The device is disconnected. • The device receives resume signalling on its upstream port. This bit is cleared when read.	0
		0	SUS bit not changed.	
		1	SUS bit changed. At the same time a DEV_STAT interrupt is generated.	
4	RST		Bus Reset bit. On a bus reset, the device will automatically go to the default state. In the default state: • Device is unconfigured. • Will respond to address 0. • Control endpoint will be in the Stalled state. • Data toggling is reset for all endpoints. • All buffers are cleared. • There is no change to the endpoint interrupt status. • DEV_STAT interrupt is generated. Note: Bus resets are ignored when the device is not connected (CON=0).	0
		0	This bit is cleared when read.	
		1	This bit is set when the device receives a bus reset. A DEV_STAT interrupt is generated.	
7:5	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.11.8 Get Device Status (Command: 0xFE, Data: read 1 byte)

The Get Device Status command returns the Device Status Register. Reading the device status returns 1 byte of data. The bit field definition is same as the Set Device Status Register as shown in Table 181.

Remark: To ensure correct operation, the DEV_STAT bit of USBDevIntSt must be cleared before executing the Get Device Status command.

10.11.9 Get Error Code (Command: 0xFF, Data: read 1 byte)

Different error conditions can arise inside the SIE. The Get Error Code command returns the last error code that occurred. The 4 least significant bits form the error code.

Table 182. Get Error Code command description

Bit	Symbol	Value	Description	Reset value
3:0	EC		Error Code.	0x0
		0000	No Error.	
		0001	PID Encoding Error.	
		0010	Unknown PID.	
		0011	Unexpected Packet - any packet sequence violation from the specification.	
		0100	Error in Token CRC.	
		0101	Error in Data CRC.	
		0110	Time Out Error.	
		0111	Babble.	
		1000	Error in End of Packet.	
		1001	Sent/Received NAK.	
		1010	Sent Stall.	
		1011	Buffer Overrun Error.	
		1100	Sent Empty Packet (ISO Endpoints only).	
		1101	Bitstuff Error.	
		1110	Error in Sync.	
		1111	Wrong Toggle Bit in Data PID, ignored data.	
4	EA	-	The Error Active bit will be reset once this register is read.	
7:5	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.11.10 Select Endpoint (Command: 0x00 - 0x09 Data: read 1 byte (optional))

The Select Endpoint command initializes an internal pointer to the start of the selected buffer in EP_RAM. Optionally, this command can be followed by a data read, which returns some additional information on the packet(s) in the endpoint buffer(s). The command code of the Select Endpoint command is equal to the physical endpoint number. In the case of a single buffered endpoint the B_2_FULL bit is not valid.

Table 183. Select Endpoint command description

Bit	Symbol	Value	Description	Reset value
0	FE		Full/Empty. This bit indicates the full or empty status of the endpoint buffer(s). For IN endpoints, the FE bit gives the ANDed result of the B_1_FULL and B_2_FULL bits. For OUT endpoints, the FE bit gives ORed result of the B_1_FULL and B_2_FULL bits. For single buffered endpoints, this bit simply reflects the status of B_1_FULL.	0
		0	For an IN endpoint, at least one write endpoint buffer is empty.	
		1	For an OUT endpoint, at least one endpoint read buffer is full.	

Table 183. Select Endpoint command description

Bit	Symbol	Value	Description	Reset value
1	ST		Stalled endpoint indicator.	0
		0	The selected endpoint is not stalled.	
		1	The selected endpoint is stalled.	
2	STP		SETUP bit: the value of this bit is updated after each successfully received packet (i.e. an ACKed package on that particular physical endpoint).	0
		0	The STP bit is cleared by doing a Select Endpoint/Clear Interrupt on this endpoint.	
		1	The last received packet for the selected endpoint was a SETUP packet.	
3	PO		Packet over-written bit.	0
		0	The PO bit is cleared by the 'Select Endpoint/Clear Interrupt' command.	
		1	The previously received packet was over-written by a SETUP packet.	
4	EPN		EP NAKed bit indicates sending of a NAK. If the host sends an OUT packet to a filled OUT buffer, the device returns NAK. If the host sends an IN token packet to an empty IN buffer, the device returns NAK.	0
		0	The EPN bit is reset after the device has sent an ACK after an OUT packet or when the device has seen an ACK after sending an IN packet.	
		1	The EPN bit is set when a NAK is sent and the interrupt on NAK feature is enabled.	
5	B_1_FULL		The buffer 1 status.	0
		0	Buffer 1 is empty.	
		1	Buffer 1 is full.	
6	B_2_FULL		The buffer 2 status.	0
		0	Buffer 2 is empty.	
		1	Buffer 2 is full.	
7	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.11.11 Select Endpoint/Clear Interrupt (Command: 0x40 - 0x47, Data: read 1 byte)

Commands 0x40 to 0x47 are identical to their Select Endpoint equivalents, with the following differences:

- They clear the bit corresponding to the endpoint in the USBEpIntSt register.
- In case of a control OUT endpoint, they clear the STP and PO bits in the corresponding Select Endpoint Register.
- Reading one byte is obligatory.

10.11.12 Set Endpoint Status (Command: 0x40 - 0x49, Data: write 1 byte (optional))

The Set Endpoint Status command sets status bits 7:5 and 0 of the endpoint. The Command Code of Set Endpoint Status is equal to the sum of 0x40 and the physical endpoint number in hex. Not all bits can be set for all types of endpoints.

Table 184. Set Endpoint Status command description

Bit	Symbol	Value	Description	Reset value
0	ST		Stalled endpoint bit. A Stalled control endpoint is automatically unstalled when it receives a SETUP token, regardless of the content of the packet. If the endpoint should stay in its stalled state, the CPU can stall it again by setting this bit. When a stalled endpoint is unstalled - either by the Set Endpoint Status command or by receiving a SETUP token - it is also re-initialized. This flushes the buffer: in case of an OUT buffer it waits for a DATA 0 PID; in case of an IN buffer it writes a DATA 0 PID. There is no change of the interrupt status of the endpoint. When already unstalled, writing a zero to this bit initializes the endpoint. When an endpoint is stalled by the Set Endpoint Status command, it is also re-initialized.	0
		0	The endpoint is unstalled.	
		1	The endpoint is stalled.	
4:1	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
5	DA		Disabled endpoint bit.	0
		0	The endpoint is enabled.	
		1	The endpoint is disabled.	
6	RF_MO		Rate Feedback Mode.	0
		0	Interrupt endpoint is in the Toggle mode.	
		1	Interrupt endpoint is in the Rate Feedback mode. This means that transfer takes place without data toggle bit.	
7	CND_ST		Conditional Stall bit.	0
		0	Unstalls both control endpoints.	
		1	Stall both control endpoints, unless the STP bit is set in the Select Endpoint register. It is defined only for control OUT endpoints.	

10.11.13 Clear Buffer (Command: 0xF2, Data: read 1 byte (optional))

When an OUT packet sent by the host has been received successfully, an internal hardware FIFO status Buffer_Full flag is set. All subsequent packets will be refused by returning a NAK. When the device software has read the data, it should free the buffer by issuing the Clear Buffer command. This clears the internal Buffer_Full flag. When the buffer is cleared, new packets will be accepted.

When bit 0 of the optional data byte is 1, the previously received packet was over-written by a SETUP packet. The Packet over-written bit is used only in control transfers. According to the USB specification, a SETUP packet should be accepted irrespective of the buffer status. The software should always check the status of the PO bit after reading the SETUP data. If it is set then it should discard the previously read data, clear the PO bit by issuing a Select Endpoint/Clear Interrupt command, read the new SETUP data and again check the status of the PO bit.

See Section 10.13 “Functional description” for a description of when this command is used.

Table 185. Clear Buffer command description

Bit	Symbol	Value	Description	Reset value
0	PO		Packet over-written bit. This bit is only applicable to the control endpoint EP0.	0
		0	The previously received packet is intact.	
		1	The previously received packet was over-written by a later SETUP packet.	
7:1	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.11.14 Validate Buffer (Command: 0xFA, Data: none)

When the CPU has written data into an IN buffer, software should issue a Validate Buffer command. This tells hardware that the buffer is ready for sending on the USB bus. Hardware will send the contents of the buffer when the next IN token packet is received.

Internally, there is a hardware FIFO status flag called Buffer_Full. This flag is set by the Validate Buffer command and cleared when the data has been sent on the USB bus and the buffer is empty.

A control IN buffer cannot be validated when its corresponding OUT buffer has the Packet Over-written (PO) bit (see the Clear Buffer Register) set or contains a pending SETUP packet. For the control endpoint the validated buffer will be invalidated when a SETUP packet is received.

See Section 10.13 “Functional description” for a description of when this command is used.

Remark: For sending an empty packet, the Validate Buffer command should also be used.

10.12 USB device controller initialization

The LPC134x USB device controller initialization includes initialization of the USB clock and the device controller.

10.12.1 USB clock configuration

1. Enable the PLL for USB clock if the PLL is used.
2. Set the system oscillator control register SYSOSCCTRL (see Table 14) to 0.
3. Enable the system oscillator by clearing bit 5 in the PDAWAKECFG register (see Table 54), then wait 200 μs for system oscillator to stabilize.

4. Select the system clock source by setting 0x01 (use system oscillator) in SYSPLLCLKSEL register (see Table 18).
5. Update the clock source by setting 1 in SYSPLLUEN register (see Table 19) register, and wait until clock source is updated.
6. Boost system PLL to 192 MHz and then divide by 4 (M=4, P=2) to obtain the 48 MHz main clock. If the main clock is not 48 MHz then the USB PLL has to be used as USB clock.
7. Enable the main system PLL by clearing bit 7 in PDAWAKECFG (see Table 54), then wait until the PLL clock is locked.
8. If the USB PLL is used as the USB clock, do the following extra step:

   Configure USB PLL identically to the System PLL and select system clock source by setting 0x01 (use system oscillator) in USBPLLCLKSEL (see Table 31) register.
9. Enable USB clock by clearing bit 8 in PDCTRL.
10. Set USB clock by setting USBCLKSEL (see Table 31) to:
    a. 0x0 if USB PLL is used.
    b. 0x1 if main clock is used.
11. Update clock source by setting 1 in USBPLLUEN (see Table 21) register and wait until USB clock source is updated.
12. Set USB clock divider register (see Table 30) to 1, meaning the USB clock is divided by 1, as the input is 48 MHz already.

10.12.2 USB device controller initialization

1. Set bits 14 and 16 in the AHBCLKCTRL register (see Table 25) to enable USB and IOConfig blocks. The IOConfig is needed to configure IO pin multiplexing.
2. In the IOConfig block, set Port0[3] (see Table 106) and Port0[6] (see Table 114) to USB VBUS and USB CONNECT respectively.
3. Clear any device interrupts using USBDevIntClr (Table 165), then enable the desired endpoints by setting the corresponding bits in USBDevIntEn (Table 164).
4. Install the USB interrupt handler in the NVIC.
5. Set the default USB address to 0x0 and DEV_EN to 1 using the SIE Set Address command.
6. Set CON bit to 1 to make CONNECT active using the SIE Set Device Status command.

10.13 Functional description

10.13.1 Data flow from the Host to the Device

The USB ATX receives the bi-directional USB_DP and USB_DM lines of the USB bus. It will put this data in the unidirectional interface between ATX and USB block.

The SIE protocol engine receives this serial data and converts it into a parallel data stream. The parallel data is sent to the RAM interface which in turn will transfer the data to the endpoint buffer. The endpoint buffer is implemented as an SRAM based FIFO. Data is written to the buffers with the header showing how many bytes are valid in the buffer.

For non-isochronous endpoints when a full data packet is received without any errors, the endpoint generates a request for data transfer from its FIFO by generating an interrupt to the system.

Isochronous endpoint will have one packet of data to be transferred in every frame. This requires the data transfer has to be synchronized to the USB frame rather than packet arrival. The 1 KHz free running clock re synchronized on the incoming SoF tokens will generate an interrupt every millisecond.

The data transfer follows the little endian format. The first byte received from the USB bus will be available in the LS byte of the receive data register.

10.13.2 Data flow from the Device to the Host

For data transfer from an endpoint to the host, the host will send an IN token to that endpoint. If the FIFO corresponding to the endpoint is empty, the device will return a NAK and will generate an interrupt (assuming the interrupt on NAK is enabled). On this interrupt the processor fills a packet of data in the endpoint FIFO. The next IN token that comes--after filling this packet--will transfer this packet to the host.

The data transfer follows the little endian format. The first byte sent on the USB bus will be the LS byte of the transmit data register.

Remark: USB is a host controlled protocol, i.e., irrespective of whether the data transfer is from the host to the device or from the device to the host, the transfer sequence is always initiated by the host. During data transfer from the device to the host, the host sends an IN token to the device, following which the device responds with the data.

10.13.3 Interrupt based transfer

Interrupt based data transfer is done through the interrupt issued from the USB core to the processor.

Reception of a valid (error-free) data packet in any of the OUT non-isochronous endpoint buffer generates an interrupt. Upon receiving the interrupt, the software can read the data using receive length and data registers. When there is no empty buffer (for a given non-isochronous OUT endpoint), any data arrival generates an interrupt only if Interrupt On NAK feature for that endpoint type is enabled and existing interrupt is cleared.

Similarly, when a packet is successfully transferred to the host from any IN non-isochronous endpoint buffer, an interrupt is generated. When there is no data available in any of the buffers (for a given non-isochronous IN endpoint), a data request generates an interrupt only if Interrupt On NAK feature for that endpoint type is enabled and existing interrupt is cleared. Upon receiving the interrupt, the software can load any data to be sent using transmit length and data registers.

10.13.4 Isochronous transfer

Isochronous endpoints are double-buffered and the buffer toggling will happen only on frame boundaries i.e., at every 1 ms. ‘Clear Buffer’ and ‘Validate Buffer’ do not cause the buffer to toggle.

For OUT isochronous endpoints, the data will always be written irrespective of the buffer status. For IN isochronous endpoints, the data available in the buffer will be sent only if the buffer is validated; otherwise, an empty packet will be sent.

There will not be any interrupt generated specific to isochronous endpoints other than the frame interrupt.

It is assumed that the Isochronous pipe is open at the reception of a request "Set Interface (alternate setting > 0)". This request is sent to the interface to which the isochronous endpoint belongs.

This means that the device is expecting the first isochronous transfer within the millisecond.

10.13.5 Automatic stall feature

The USB block includes a Hardware STALL mechanism. H/W STALL will occur in

the following control transactions:

- Data stage consists of INs, the status is a single OUT transaction with an empty packet sent by the host.
- Data stage consists of OUTs, the status is a single IN transaction, for which the device respond with an empty packet.
- Setup stage followed by a Status stage consisting of an IN transaction, for which the device respond with empty packet.

A STALL will **not** occur in the following situations:

- Data stage consists of OUTs, the status is a single IN transaction, for which the device respond with a non-empty packet.
- Setup stage followed by a Status stage consisting of an IN transaction, for which the device respond with a non-empty packet.

10.14 Double-buffered endpoint operation

The Bulk and Isochronous endpoints of the USB Device Controller are double-buffered to increase data throughput.

For the following discussion, the endpoint buffer currently accessible to the CPU for reading or writing is said to be the active buffer.

10.14.1 Bulk endpoints

For Bulk endpoints, the active endpoint buffer is switched by the SIE Clear Buffer or Validate Buffer commands.

The following example illustrates how double-buffering works for a Bulk OUT endpoint in Slave mode:

Assume that both buffer 1 (B_1) and buffer 2 (B_2) are empty, and that the active buffer is B_1.

1. The host sends a data packet to the endpoint. The device hardware puts the packet into B_1, and generates an endpoint interrupt.
2. Software clears the endpoint interrupt and begins reading the packet data from B_1. While B_1 is still being read, the host sends a second packet, which device hardware places in B_2, and generates an endpoint interrupt.
3. Software is still reading from B_1 when the host attempts to send a third packet. Since both B_1 and B_2 are full, the device hardware responds with a NAK.
4. Software finishes reading the first packet from B_1 and sends a SIE Clear Buffer command to free B_1 to receive another packet. B_2 becomes the active buffer.
5. Software sends the SIE Select Endpoint command to read the Select Endpoint Register and test the FE bit. Software finds that the active buffer (B_2) has data (FE=1). Software clears the endpoint interrupt and begins reading the contents of B_2.
6. The host resends the third packet which device hardware places in B_1. An endpoint interrupt is generated.
7. Software finishes reading the second packet from B_2 and sends a SIE Clear Buffer command to free B_2 to receive another packet. B_1 becomes the active buffer. Software waits for the next endpoint interrupt to occur (it already has been generated back in step 6).
8. Software responds to the endpoint interrupt by clearing it and begins reading the third packet from B_1.
9. Software finishes reading the third packet from B_1 and sends a SIE Clear Buffer command to free B_1 to receive another packet. B_2 becomes the active buffer.
10. Software tests the FE bit and finds that the active buffer (B_2) is empty (FE=0).
11. Both B_1 and B_2 are empty. Software waits for the next endpoint interrupt to occur. The active buffer is now B_2. The next data packet sent by the host will be placed in B_2.

The following example illustrates how double-buffering works for a Bulk IN endpoint in Slave mode:

Assume that both buffer 1 (B_1) and buffer 2 (B_2) are empty and that the active buffer is B_1. The interrupt on NAK feature is enabled.

1. The host requests a data packet by sending an IN token packet. The device responds with a NAK and generates an endpoint interrupt.
2. Software clears the endpoint interrupt. The device has three packets to send. Software fills B_1 with the first packet and sends a SIE Validate Buffer command. The active buffer is switched to B_2.
3. Software sends the SIE Select Endpoint command to read the Select Endpoint Register and test the FE bit. It finds that B_2 is empty (FE=0) and fills B_2 with the second packet. Software sends a SIE Validate Buffer command, and the active buffer is switched to B_1.
4. Software waits for the endpoint interrupt to occur.
5. The device successfully sends the packet in B_1 and clears the buffer. An endpoint interrupt occurs.

6. Software clears the endpoint interrupt. Software fills B_1 with the third packet and validates it using the SIE Validate Buffer command. The active buffer is switched to B_2.
7. The device successfully sends the second packet from B_2 and generates an endpoint interrupt.
8. Software has no more packets to send, so it simply clears the interrupt.
9. The device successfully sends the third packet from B_1 and generates an endpoint interrupt.
10. Software has no more packets to send, so it simply clears the interrupt.
11. Both B_1 and B_2 are empty, and the active buffer is B_2. The next packet written by software will go into B_2.

10.14.2 Isochronous endpoints

For isochronous endpoints, the active data buffer is switched by hardware when the FRAME interrupt occurs. The SIE Clear Buffer and Validate Buffer commands do not cause the active buffer to be switched.

Double-buffering allows the software to make full use of the frame interval writing or reading a packet to or from the active buffer, while the packet in the other buffer is being sent or received on the bus.

For an OUT isochronous endpoint, any data not read from the active buffer before the end of the frame is lost when it switches.

For an IN isochronous endpoint, if the active buffer is not validated before the end of the frame, an empty packet is sent on the bus when the active buffer is switched, and its contents will be overwritten when it becomes active again.

UM10375

Chapter 11: LPC13xx USB on-chip drivers

Rev. 5 — 21 June 2012 User manual

11.1 How to read this chapter

The USB device controller is available on parts LPC1342 and LPC1343 only.

11.2 Introduction

The boot ROM contains a USB driver to simplify the USB application development. The USB driver implements the Human Interface Device (HID) and the Mass Storage Device (MSC) device class. Only one device function, either HID or MSC, can be used by the application software. The USB enumeration and commands are handled by the boot ROM code. The application software only needs to provide the callback functions to handle the data sent or requested by the host.

11.3 USB driver functions

The following four functions of the USB driver software are exposed to the user application:

1. Clock and pin initialization
2. USB initialization
3. USB connect
4. USB interrupt handler

11.3.1 Clock and pin initialization

This function configures the LPC134x assuming an external crystal with 12 MHz clock frequency:

- The system PLL is configured to output a 48 MHz clock.
- The main clock is connected to the system PLL output, and the input to the USB clock divider is connected to the main clock.
- The USB bit in the AHB clock divider is set to 1.
- The USB pins are connected to the USB block, and the USB PHY is enabled.
- The USB clock divider is set to 1.
- The USB PLL is enabled.

Calling this function is optional if application software implements an equivalent initialization routine which includes setting up the USB clock to 48 MHz and connecting USB_DM and USB_DP pins to the USB peripheral. After reset the USB_DM and USB_DP pins are connected to the GPIO peripheral.

11.3.2 USB initialization

This function must be called by the application software after the clock and pin initialization. A pointer to the structure describing the USB device type is passed as a parameter to this function. The USB device type can be HID or MSC. The pointer is stored for future reference. The USB device controller is initialized and the corresponding USB interrupt channel is enabled in the NVIC.

11.3.3 USB connect

This function is called after the USB initialization. This function uses the soft connect feature to make the device visible on the USB bus. This function is called only after the application is ready to handle the USB data. The enumeration process is started by the host after the device detection. The driver handles the enumeration process according to the USB device type pointer passed in the USB initialization function.

11.3.4 USB interrupt handler

When the user application is active the interrupt handlers are mapped in the user flash space. The user application must provide an interrupt handler for the USB interrupt and call this function in the interrupt handler routine. The driver interrupt handler takes appropriate action according to the data received on the USB bus and the configured device type (HID or MSC).

11.4 Calling the USB device driver

A fixed location in ROM contains a pointer to the ROM driver table i.e. 0x1FFF 1FF8. This location is kept the same for a device family. The ROM driver table contains a pointer to the USB driver table. Pointers to the various USB driver functions are stored in this table. USB driver functions can be called by using a C structure. Figure 19 illustrates the pointer mechanism used to access the on-chip USB driver.

On-chip RAM from address 0x1000 0050 to 0x1000 0180 is used by the USB driver. This address range should not be used by the application. For applications using the on-chip USB driver, the linker control file should be modified appropriately to prevent usage of this area for the application’s variable storage

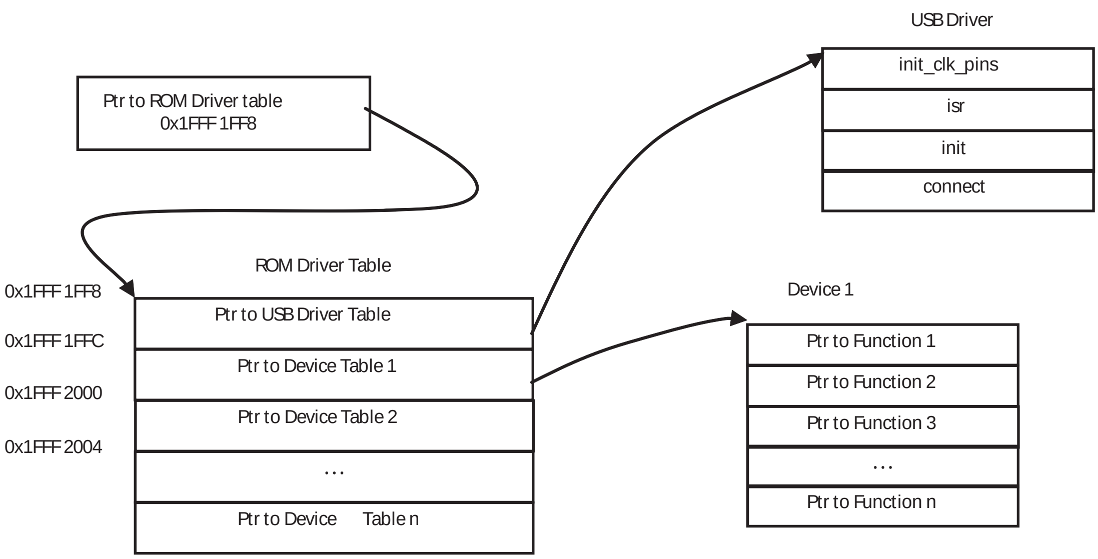


Fig 19. USB device driver pointer structure

11.4.1 USB mass storage driver

The following steps illustrate the USB mass storage driver usage. A complete example is available in the LPC13xx code bundle.

1. Map the pointer to the on chip driver table:

```
ROM ** rom = (ROM **) 0x1fff1ff8;
```

2. Enable 32-bit timer 1 (CT32B1) and IOCONFIG block:

```
LPC_SYSCON->SYSAHBCLKCTRL |= (EN_TIMER32_1 | EN_IOCON);
```

3. Initialize USB clock and pins:

```
(*rom)->pUSBD->init_clk_pins();
```

4. Set up device type and information:

```
USB_DEV_INFO DeviceInfo;
MSC_DEVICE_INFO MscDevInfo;
MscDevInfo.idVendor = USB_VENDOR_ID;
MscDevInfo.idProduct = USB_PROD_ID;
MscDevInfo.bcdDevice = USB_DEVICE;
MscDevInfo.StrDescPtr = (uint32_t)&USB_StringDescriptor[0];
MscDevInfo.MSCInquiryStr = (uint32_t)&InquiryStr[0];
MscDevInfo.BlockSize = MSC_BlockSize;
MscDevInfo.BlockCount = MSC_BlockCount;
MscDevInfo.MemorySize = MSC_MemorySize;
MscDevInfo.MSC_Read = MSC_MemoryRead;
MscDevInfo.MSC_Write = MSC_MemoryWrite;
```

5. Initialize the USB:

```
DeviceInfo.DevType = USB_DEVICE_CLASS_STORAGE;
DeviceInfo.DevDetailPtr = (uint32_t)&MscDevInfo;
(*rom)->pUSBD->init(&DeviceInfo);
```

6. Initialize the mass storage state machine:

```
(uint32_t *) BulkStage = 0x10000054;
*BulkStage = 0x0;
```

7. Add the USB interrupt handler to your project:

```
USB_IRQHandler(void)
{
  (*rom)->pUSBD->isr();
}
```

8. Call USB connect:

```
(*rom)->pUSBD->connect(TRUE);
```

11.4.2 USB human interface driver

The following steps show how to use the USB human interface driver. A complete example is available in the LPC13xx code bundle.

1. Map the pointer to the on chip driver table:

```
ROM ** rom = (ROM **)0x1fff1ff8;
```

2. Enable 32-bit timer 1 (CT32B1) and IOCONFIG block:

```
LPC_SYSCON->SYSAHBCLKCTRL |= (EN_TIMER32_1 | EN_IOCON);
```

3. Initialize USB clock and pins:

```
(*rom)->pUSBD->init_clk_pins();
```

4. Set up device type and information:

```
USB_DEV_INFO DeviceInfo;
HID_DEVICE_INFO HidDevInfo;
HidDevInfo.idVendor = USB_VENDOR_ID;
HidDevInfo.idProduct = USB_PROD_ID;
HidDevInfo.bcdDevice = USB_DEVICE;
HidDevInfo.StrDescPtr = (uint32_t)&USB_StringDescriptor[0];
HidDevInfo.InReportCount = 1;
HidDevInfo.OutReportCount = 1;
HidDevInfo.SampleInterval = 0x20;
HidDevInfo.InReport = GetInReport;
HidDevInfo.OutReport = SetOutReport;
```

5. Initialize the USB:

```
DeviceInfo.DevType = USB_DEVICE_CLASS_HUMAN_INTERFACE;
DeviceInfo.DevDetailPtr = (uint32_t)&HidDevInfo;
(*rom)->pUSBD->init(&DeviceInfo);
```

6. Add the USB interrupt handler to your project:

```
USB_IRQHandler(void)
{
  (*rom)->pUSBD->isr();
}
```

7. Call USB connect:

```
USB Connect
     (*rom)->pUSBD->connect(TRUE);
```

11.5 USB driver structure definitions

11.5.1 ROM driver table

The following structure is used to access the USB driver table stored in ROM:

```
typedefstruct _ROM {
   const USBD * pUSBD;
}  ROM;
```

11.5.2 USB driver table

The following structure is used to access the functions exposed by the USB driver:

```
typedefstruct _USBD {
  void (*init_clk_pins)(void);
  void (*isr)(void);
  void (*init)( USB_DEV_INFO * DevInfoPtr );
  void (*connect)(uint32_t  con);
}  USBD;
```

11.5.3 USB device information

The following structure is used to pass the USB device type and information:

```
typedef struct _USB_DEVICE_INFO {
  uint16_t DevType;
  uint32_t  DevDetailPtr;
} USB_DEV_INFO;
```

Table 186. USB device information class structure

Member	Description

DevType	USB device class type
	USB_DEVICE_CLASS_HUMAN_INTERFACE(0x03)
	USB_DEVICE_CLASS_STORAGE(0x08)

DevDetailPtr	Pointer to the device information structure

11.5.4 Mass storage device information

The following structure is used to pass the MSC device information:

```
typedefstruct _MSC_DEVICE_INFO {
  uint16_t   idVendor;
  uint16_t   idProduct;
  uint16_t   bcdDevice;
  uint32_t  StrDescPtr;
  uint32_t  MSCInquiryStr;
  uint32_t  BlockCount;
  uint32_t  BlockSize;
  uint32_t  MemorySize;
  void (*MSC_Write)( uint32_t offset, uint8_t src[], uint32_t length);
  void (*MSC_Read)( uint32_t offset, uint8_t dst[], uint32_t length);
}  MSC_DEVICE_INFO;
```

Table 187. Mass storage device information class structure

Member	Description
idVendor	Vendor ID
idProduct	Product ID
bcdDevice	Device release number
StrDescPtr	Pointer to the String Describing the Manufacturer, Product and Serial number. Refer to Section 11.6.4 for an example.
MSCInquiryStr	Pointer to the 28 character string. This string is sent in response to the SCSI Inquiry command
BlockCount	Number of blocks present in the mass storage device
BlockSize	Block size in number of bytes

Table 187. Mass storage device information class structure

Member	Description
MemorySize	Memory size in number of bytes
MSC_Write	Write call back function. This function is provided by the application software. This function gets called when host sends a write command. Input Parameters: Offset – Destination start address Source Pointer – Pointer to the source of data Length – Number of bytes to be written
MSC_Read	Read call back function. This function is provided by the application software. This function gets called when host sends a read command. Input Parameters: Offset – Destination start address Destination Pointer – Pointer to the destination of data Length – Number of bytes to be read

11.5.5 Human interface device information

The following structure is used to pass the HID device information:

```
typedefstruct _HID_DEVICE_INFO {
  uint16_t   idVendor;
  uint16_t   idProduct;
  uint16_t   bcdDevice;
  uint32_t  StrDescPtr;
  uint8_t   InReportCount;
  uint8_t   OutReportCount;
  uint8_t   SampleInterval;
  void (*InReport)( uint8_t src[], uint32_t length);
  void (*OutReport)(uint8_t dst[], uint32_t length);
}  HID_DEVICE_INFO;
```

Table 188. Human interface device information class structure

Member	Description
idVendor	Vendor ID
idProduct	Product ID
bcdDevice	Device release number
StrDescPtr	Pointer to the String Describing the Manufacturer, Product and Serial number. Refer to Section 11.6.4 for an example.
InReportCount	Number of bytes in InReport
OutReportCount	Number of bytes in OutReport

Table 188. Human interface device information class structure

Member	Description
SampleInterval	Interrupt endpoint (IN and OUT) sample interval in ms.
InReport	InReport call back function. This function is provided by the application software. This function gets called when host sends a InReport command. Input Parameters: Source Pointer – Pointer to the destination of data Length – Number of bytes to be read
OutReport	OutReport call back function. This function is provided by the application software. This function gets called when host sends a OutReport command. Input Parameters: Source Pointer – Pointer to the source of data Length – Number of bytes to be written

11.6 USB descriptors

The USB driver reports several predefined descriptors during enumeration of the HID or MSC device types. Certain fields of the descriptor are user-defined.

11.6.1 Standard descriptor

The USB driver reports the following predefined descriptor during enumeration of the HID or MSC device. Fields in bold are user-defined.

Standard descriptors are combined in one string and the offset is fixed. iManufacture, iProduct, and iSerialNumber strings must be of fixed size. See Section 11.6.4 for HID and MSC descriptor examples.

Table 189. Standard descriptor

Field	Value	Description
bLength	0x12	Descriptor size in bytes
bDescriptorType	0x01	The constant Device (01h)
bcdUSB	0x0200	USB Specification release number (BCD)
bDeviceClass	0x00	Class Code
bDeviceSubClass	0x00	Subclass code
bDeviceProtocol	0x00	Protocol Code
bMaxPacketSize0	0x40	Maximum Packet size for endpoint 0
idVendor	User Defined	Vendor ID (Provided by the Application Software)
idProduct	User Defined	Product ID (Provided by the Application Software)
bcdDevice	User Defined	Device Release number
iManufacturer	0x04	Index of string descriptor for the manufacturer (fixed size)
iProduct	0x20	Index of string descriptor for the product (fixed size)
iSerialNumber	0x48	Index of string descriptor containing the serial number (fixed size)
bNumConfigurations	0x01	Number of possible configurations

11.6.2 Mass storage configuration, interface, and endpoint descriptors

The USB driver reports the following descriptors during the enumeration process for a MSC device:

Table 190. Mass storage descriptors

Field	Value	Description
Mass storage configuration descriptor		
bLength	0x09	Descriptor size in bytes
bDescriptorType	0x02	The constant Configuration (0x02)
wTotalLength	0x0020	Size of all data returned for this configuration in bytes
bNumInterfaces	0x01	Number of interfaces the configuration supports
bConfigurationValue	0x01	Identifier for Set_Configuration and Get_Configuration requests
iConfiguration	0x00	Index of string descriptor for the configuration
bmAttributes	0xc0	Self/Bus power, Remote wakeup
bMaxPower	0x32	Bus Power required, expressed as (max mA/2)
Mass storage interface descriptor		
bLength	0x09	Descriptor size in bytes
bDescriptorType	0x04	The constant Interface (0x04)
bInterfaceNumber	0x00	Number identifying this interface
bAlternateSetting	0x00	Value used to select an alternate setting
bNumEndpoints	0x02	Number of endpoints supported
bInterfaceClass	0x08	Class code, Storage
bInterfaceSubClass	0x06	SubClass code, SCSI
bInterfaceProtocol	0x50	Protocol code, Bulk only
iInterface	0x62	Index of string descriptor for the interface
Mass storage bulk IN endpoint descriptor		
bLength	0x07	Descriptor size in bytes
bDescriptorType	0x05	The constant Endpoint (0x05)
bEndpointAddress	0x82	Endpoint number and direction
bmAttributes	0x02	Transfer type supported, Bulk
wMaxPacketSize	0x0040	Maximum packet size supported
bInterval	0x00	Maximum latency/polling interval/NAK rate
Mass storage bulk OUT endpoint descriptor		
bLength	0x07	Descriptor size in bytes
bDescriptorType	0x05	The constant Endpoint (0x05)
bEndpointAddress	0x02	Endpoint number and direction
bmAttributes	0x02	Transfer type supported, Bulk
wMaxPacketSize	0x0040	Maximum packet size supported
bInterval	0x00	Maximum latency/polling interval/NAK rate

11.6.3 HID configuration, interface, class, endpoint, and report descriptor

The USB driver reports the following descriptors during the enumeration process for an HID device:

Table 191. HID descriptors

Field	Value	Description
HID configuration descriptor		
bLength	0x09	Descriptor size in bytes
bDescriptorType	0x02	The constant Configuration (0x02)
wTotalLength	0x0029	Size of all data returned for this configuration in bytes
bNumInterfaces	0x01	Number of interfaces the configuration supports
bConfigurationValue	0x01	Identifier for Set_Configuration and Get_Configuration requests
iConfiguration	0x00	Index of string descriptor for the configuration
bmAttributes	0xc0	Self/Bus power, Remote wakeup
bMaxPower	0x32	Bus Power required, expressed as (max mA/2)
HID interface descriptor		
bLength	0x09	Descriptor size in bytes
bDescriptorType	0x04	The constant Interface (0x04)
bInterfaceNumber	0x00	Number identifying this interface
bAlternateSetting	0x00	Value used to select an alternate setting
bNumEndpoints	0x02	Number of endpoints supported
bInterfaceClass	0x03	Class code, Human Interface
bInterfaceSubClass	0x00	Sublass code, None
bInterfaceProtocol	0x00	Protocol code, None
iInterface	0x62	Index of string descriptor for the interface
HID class descriptor		
bLength	0x09	Descriptor size in bytes
bDescriptorType	0x21	The constant HID class (0x21)
bcdHID	0x0100	HID specification release number (BCD)
bCountryCode	0x00	Country identifier
bNumDescriptors	0x01	Number of subordinate class descriptors supported
bDescriptorType	0x22	The type of class descriptor, HID Report
wDescriptorLength	0x001b	Total length of report descriptor
HID interrupt IN descriptor		
bLength	0x07	Descriptor size in bytes
bDescriptorType	0x05	The constant Endpoint (0x05)
bEndpointAddress	0x81	Endpoint number and direction
bmAttributes	0x03	Transfer type supported, Interrupt
wMaxPacketSize	0x0040	Maximum packet size supported
bInterval	User Defined	Polling interval (Provided by the Application Software)
HID interrupt OUT descriptor		
bLength	0x07	Descriptor size in bytes
bDescriptorType	0x05	The constant Endpoint (0x05)
bEndpointAddress	0x01	Endpoint number and direction
bmAttributes	0x03	Transfer type supported, Interrupt
wMaxPacketSize	0x0040	Maximum packet size supported

Table 191. HID descriptors

Field	Value	Description
bInterval	User Defined	Polling interval (Provided by the Application Software)
HID report descriptor		
Usage Page	0x06 01 00	Generic Desktop
Usage	0x09 01	Vendor Usage 1
Collection	0xA1 01	Start of Collection, Application
Logical Minimum	0x15 00	Minimum value 0x00
Logical Maximum	0x26 FF 00	Maximum Value 0xFF
Report Size	0x75 08	Number of bits, 8
Report Count	0x95 xx	Provided by the application software
Usage	0x09 01	Vendor usage 1
Input	0x81 02	Data, Variable, Absolute
Report Count	0x95 xx	Provided by the application software
Usage	0x09 01	Vendor usage 1
Output	0x91 02	Data, Variable, Absolute
End Collection	0xC0	End of collection

11.6.4 Example descriptors

Example HID descriptor

```
USB_HID_StringDescriptor[] = {  0x04, USB_STRING_DESCRIPTOR_TYPE, 0x0409

/* Index 0x04: Manufacturer */

  0x1C, USB_STRING_DESCRIPTOR_TYPE,

  'N',0, 'X',0, 'P',0,' ',0,'S',0, 'E',0, 'M',0,'I',0, 'C',0,'O',0,'N',0,‘D',0,' ',0,

/* Index 0x20: Product */

  0x28,USB_STRING_DESCRIPTOR_TYPE,        /* bDescriptorType */

  'N',0,  'X',0,  'P',0,  ' ',0,  'L',0,  'P',0,  'C',0,  '1',0,  '3',0,  'X',0,
     'X',0,  ' ',0,  'H',0,  'I',0,  'D',0,  ' ',0,  ' ',0,  ' ',0,  ' ',0,

/* Index 0x48: Serial Number */

  0x1A, USB_STRING_DESCRIPTOR_TYPE,        /* bDescriptorType */

  'D',0,  'E',0,  'M',0,  'O',0,  '0',0,  '0',0,  '0',0,  '0',0,  '0',0,  '0',0,
     '0',0,  '0',0,

/* Index 0x62: Interface 0, Alternate Setting 0 */

  0x0E, USB_STRING_DESCRIPTOR_TYPE,        /* bDescriptorType */

  'H',0, 'I',0,  'D',0,  ' ',0,  ' ',0,  ' ',0,

};
```

Example MSC descriptor

```
USB_ISP_StringDescriptor[] = {0x04,USB_STRING_DESCRIPTOR_TYPE,0x0409,

/* Index 0x04: Manufacturer */

  0x1C,USB_STRING_DESCRIPTOR_TYPE,

  'N',0,'X',0,'P',0,' ',0,'S',0,'e',0,'m',0,'i',0,'c',0,'o',0,'n',0,'d',0,' ',0,

/* Index 0x20: Product */

  0x28, USB_STRING_DESCRIPTOR_TYPE,

  'N',0,'X',0,'P',0,' ',0,'L',0,'P',0,'C',0,'1',0,'3',0,'X',0,'X',0, ' ',0,'I',0,
     'F',0, 'L',0, 'A',0, 'S',0,'H',0,' ',0,

/* Index 0x48: Serial Number */

  0x1A,USB_STRING_DESCRIPTOR_TYPE,

  'I',0,'S',0,'P',0,'0',0,'0',0,'0',0,'0',0,'0',0,'0',0,'0',0, '0',0,'0',0,

/* Index 0x62: Interface 0, Alternate Setting 0 */

  0x0E,USB_STRING_DESCRIPTOR_TYPE,

  'M',0,

  'e',0,

  'm',0,

  'o',0,

  'r',0,

  'y',0,

};
```

UM10375

Chapter 12: LPC13xx UART

Rev. 5 — 21 June 2012 **User manual**

12.1 How to read this chapter

The UART block is identical for all LPC13xx parts. The $\overline{\text{DSR}}$, $\overline{\text{DCD}}$, and $\overline{\text{RI}}$ modem signals are pinned out for the LQFP48 packages only.

12.2 Basic configuration

The UART is configured using the following registers:

1. Pins: For the LPC1311/13/42/43 parts, the UART pins must be configured in the IOCONFIG register block (Section 7.4) before the UART clocks can be enabled. For the LPC1311/01 and LPC1313/01 parts, no special enabling sequence is required.

   Remark: For the LPC1311/01 and LPC1313/01 parts, the modem functions on pins PIO3_1 to PIO3_3 must be configured in the corresponding IOCONFIG registers and also in the IOCON_DSR_LOC, IOCON_DCD_LOC, and IOCON_RI_LOC registers (see Table 140 to Table 142).
2. Power: In the SYSAHBCLKCTRL register, set bit 12 (Table 25).
3. Peripheral clock: Enable the UART peripheral clock by writing to the UARTCLKDIV register (Table 27).

12.3 Features

- 16-byte receive and transmit FIFOs.
- Register locations conform to ‘550 industry standard.
- Receiver FIFO trigger points at 1, 4, 8, and 14 bytes.
- 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.

12.4 Pin description

Table 192. UART pin description

Pin	Type	Description
RXD	Input	**Serial Input.** Serial receive data.
TXD	Output	**Serial Output.** Serial transmit data.
$\overline{\text{RTS}}$	Output	Request To Send. RS-485 direction control pin.
$\overline{\text{DTR}}$	Output	Data Terminal Ready.
$\overline{\text{DSR}}$[1]	Input	Data Set Ready.

Table 192. UART pin description

Pin	Type	Description
$\overline{\text{CTS}}$	Input	Clear To Send.
$\overline{\text{DCD}}$[1]	Input	Data Carrier Detect.
$\overline{\text{RI}}$[1]	Input	Ring Indicator.

[1] LQFP48 packages only.

12.5 Clocking and power control

The clocks and power to the UART block are controlled by two registers:

1. The UART block can be enabled or disabled through the System AHB clock control register bit 12 (see Table 25).
2. The UART peripheral clock UART_PCLK is enabled in the UART clock divider register (see Table 27). This clock is used by the UART baud rate generator.

Remark: For LPC1311/13/42/43 parts, the UART pins must be configured in the corresponding IOCON registers **before** the UART clocks are enabled. For the LPC1311/01 and LPC1313/01 parts, no special enabling sequence is required.

12.6 Register description

The UART contains registers organized as shown in Table 193. The Divisor Latch Access Bit (DLAB) is contained in U0LCR[7] and enables access to the Divisor Latches.

Table 193. Register overview: UART (base address: 0x4000 8000)

Name	Access	Address offset	Description	Reset value[1]
U0RBR	RO	0x000	Receiver Buffer Register. Contains the next received character to be read. When DLAB=0.	NA
U0THR	WO	0x000	Transmit Holding Register. The next character to be transmitted is written here. When DLAB=0.	NA
U0DLL	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. When DLAB=1.	0x01
U0DLM	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. When DLAB=1.	0x00
U0IER	R/W	0x004	Interrupt Enable Register. Contains individual interrupt enable bits for the 7 potential UART interrupts. When DLAB=0.	0x00
U0IIR	RO	0x008	Interrupt ID Register. Identifies which interrupt(s) are pending.	0x01
U0FCR	WO	0x008	FIFO Control Register. Controls UART FIFO usage and modes.	0x00
U0LCR	R/W	0x00C	Line Control Register. Contains controls for frame formatting and break generation.	0x00
U0MCR	R/W	0x010	Modem control register	0x00
U0LSR	RO	0x014	Line Status Register. Contains flags for transmit and receive status, including line errors.	0x60
U0MSR	RO	0x018	Modem status register	0x00
U0SCR	R/W	0x01C	Scratch Pad Register. Eight-bit temporary storage for software.	0x00
U0ACR	R/W	0x020	Auto-baud Control Register. Contains controls for the auto-baud feature.	0x00
-	-	0x024	Reserved	-
U0FDR	R/W	0x028	Fractional Divider Register. Generates a clock input for the baud rate divider.	0x10
-	-	0x02C	Reserved	-
U0TER	R/W	0x030	Transmit Enable Register. Turns off UART transmitter for use with software flow control.	0x80
-	-	0x034 - 0x048	Reserved	-
U0RS485CTRL	R/W	0x04C	RS-485/EIA-485 Control. Contains controls to configure various aspects of RS-485/EIA-485 modes.	0x00
U0RS485ADR MATCH	R/W	0x050	RS-485/EIA-485 address match. Contains the address match value for RS-485/EIA-485 mode.	0x00
U0RS485DLY	R/W	0x054	RS-485/EIA-485 direction control delay.	0x00

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

12.6.1 UART Receiver Buffer Register (U0RBR - 0x4000 8000, when DLAB = 0, Read Only)

The U0RBR 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 U0LCR must be zero in order to access the U0RBR. The U0RBR is always Read Only.

Since PE, FE and BI bits (see Table 205) 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 U0LSR register, and then to read a byte from the U0RBR.

Table 194. UART Receiver Buffer Register (U0RBR - 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.	-
31:8	-	Reserved	-

12.6.2 UART Transmitter Holding Register (U0THR - 0x4000 8000 when DLAB = 0, Write Only)

The U0THR 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 U0LCR must be zero in order to access the U0THR. The U0THR is always Write Only.

Table 195. UART Transmitter Holding Register (U0THR - 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.	-
31:8	-	Reserved	-

12.6.3 UART Divisor Latch LSB and MSB Registers (U0DLL - 0x4000 8000 and U0DLM - 0x4000 8004, 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 U0DLL and U0DLM registers together form a 16-bit divisor where U0DLL contains the lower 8 bits of the divisor and U0DLM 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 U0LCR must be one in order to access the UART Divisor Latches. Details on how to select the right value for U0DLL and U0DLM can be found in Section 12.6.15.

Table 196. UART Divisor Latch LSB Register (U0DLL - 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 U0DLM register, determines the baud rate of the UART.	0x01
31:8	-	Reserved	-

Table 197. UART Divisor Latch MSB Register (U0DLM - 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 U0DLL register, determines the baud rate of the UART.	0x00
31:8	-	Reserved	-

12.6.4 UART Interrupt Enable Register (U0IER - 0x4000 8004, when DLAB = 0)

The U0IER is used to enable the four UART interrupt sources.

Table 198. UART Interrupt Enable Register (U0IER - address 0x4000 8004 when DLAB = 0) bit description

Bit	Symbol	Value	Description	Reset value
0	RBRIE		Interrupt Enable. Enables the Receive Data Available interrupt for UART. It also controls the Character Receive Time-out interrupt.	0
		0	Disable the RDA interrupt.	
		1	Enable the RDA interrupt.	
1	THREIE		Interrupt Enable. Enables the THRE interrupt for UART. The status of this interrupt can be read from U0LSR[5].	0
		0	Disable the THRE interrupt.	
		1	Enable the THRE interrupt.	
2	RXLIE		Line Interrupt Enable. Enables the UART RX line status interrupts. The status of this interrupt can be read from U0LSR[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.	-
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 198. UART Interrupt Enable Register (U0IER - address 0x4000 8004 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.	-

12.6.5 UART Interrupt Identification Register (U0IIR - 0x4004 8008, Read Only)

U0IIR provides a status code that denotes the priority and source of a pending interrupt. The interrupts are frozen during a U0IIR access. If an interrupt occurs during a U0IIR access, the interrupt is recorded for the next U0IIR access.

Table 199. UART Interrupt Identification Register (U0IIR - address 0x4004 8008, Read Only) bit description

Bit	Symbol	Value	Description	Reset value
0	INTSTATUS		Interrupt status. Note that U0IIR[0] is active low. The pending interrupt can be determined by evaluating U0IIR[3:1].	1
		0	At least one interrupt is pending.	
		1	No interrupt is pending.	
3:1	INTID		Interrupt identification. U0IER[3:1] identifies an interrupt corresponding to the UART Rx FIFO. All other combinations of U0IER[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.	-
7:6	FIFOEN		These bits are equivalent to U0FCR[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.	-

Bits U0IIR[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 one and 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 200. Given the status of U0IIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The U0IIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The UART RLS interrupt (U0IIR[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 U0LSR[4:1]. The interrupt is cleared upon a U0LSR read.

The UART RDA interrupt (U0IIR[3:1] = 010) shares the second level priority with the CTI interrupt (U0IIR[3:1] = 110). The RDA is activated when the UART Rx FIFO reaches the trigger level defined in U0FCR7:6 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 (U0IIR[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 200. UART Interrupt Handling

U0IIR[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]	U0LSR Read[2]
0100	Second	RX Data Available	Rx data available or trigger level reached in FIFO (U0FCR0=1)	U0RBR Read[3] or UART FIFO drops below trigger level

Table 200. UART Interrupt Handling

U0IIR[3:0] value[1]	Priority	Interrupt type	Interrupt source	Interrupt reset
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	U0RBR Read[3]
0010	Third	THRE	THRE[2]	U0IIR Read[4] (if source of interrupt) or THR write
0000	Fourth	Modem status	CTS or DSR or RI or DCD	MSR read

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

[2] For details see Section 12.6.9 “UART Line Status Register (U0LSR - 0x4000 8014, Read Only)”

[3] For details see Section 12.6.1 “UART Receiver Buffer Register (U0RBR - 0x4000 8000, when DLAB = 0, Read Only)”

[4] For details see Section 12.6.5 “UART Interrupt Identification Register (U0IIR - 0x4004 8008, Read Only)” and Section 12.6.2 “UART Transmitter Holding Register (U0THR - 0x4000 8000 when DLAB = 0, Write Only)”

The UART THRE interrupt (U0IIR[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 U0THR at one time since the last THRE = 1 event. This delay is provided to give the CPU time to write data to U0THR 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 U0THR is empty. The THRE interrupt is reset when a U0THR write occurs or a read of the U0IIR occurs and the THRE is the highest interrupt (U0IIR[3:1] = 001).

It is the lowest priority interrupt and is activated whenever there is any state change on modem inputs pins, DCD, DSR or CTS. In addition, a low to high transition on modem input RI will generate a modem interrupt. The source of the modem interrupt can be determined by examining MSR[3:0]. A MSR read will clear the modem interrupt.

12.6.6 UART FIFO Control Register (U0FCR - 0x4000 8008, Write Only)

The U0FCR controls the operation of the UART RX and TX FIFOs.

Table 201. UART FIFO Control Register (U0FCR - 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 U0FCR[7:1] access. This bit must be set for proper UART operation. Any transition on this bit will automatically clear the UART FIFOs.	
1	RXFIFOR		RX FIFO Reset	0
		0	No impact on either of UART FIFOs.	
		1	Writing a logic 1 to U0FCR[1] will clear all bytes in UART Rx FIFO, reset the pointer logic. This bit is self-clearing.	
2	TXFIFOR		TX FIFO Reset	0
		0	No impact on either of UART FIFOs.	
		1	Writing a logic 1 to U0FCR[2] will clear all bytes in UART TX FIFO, reset the pointer logic. This bit is self-clearing.	
3	-		Reserved	0
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
7:6	RXTLVL		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	-

12.6.7 UART Line Control Register (U0LCR - 0x4000 800C)

The U0LCR determines the format of the data character that is to be transmitted or received.

Table 202. UART Line Control Register (U0LCR - 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 U0LCR[1:0]=00).	

Table 202. UART Line Control Register (U0LCR - address 0x4000 800C) bit description

Bit	Symbol	Value	Description	Reset Value
3	PE		Parity Enable	0
		0	Disable parity generation and checking.	
		1	Enable parity generation and checking.	
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 U0LCR[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	-

12.6.8 UART Modem Control Register

The U0MCR enables the modem loopback mode and controls the modem output signals.

Table 203. UART0 Modem Control Register (U0MCR - address 0x4000 8010) bit description

Bit	Symbol	Value	Description	Reset value
0	DTRCTRL		Source for modem output pin, DTR. This bit reads as 0 when modem loopback mode is active.	0
1	RTSCTRL		Source for modem output pin $\overline{\text{RTS}}$. 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

Table 203. UART0 Modem Control Register (U0MCR - address 0x4000 8010) bit description

Bit	Symbol	Value	Description	Reset value
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 four modem inputs (CTS, DSR, RI and DCD) are disconnected externally. Externally, the modem outputs (RTS, DTR) are set inactive. Internally, the four modem outputs are connected to the four modem inputs. As a result of these connections, the upper four bits of the U0MSR will be driven by the lower four bits of the U0MCR rather than the four modem inputs in normal mode. This permits modem status interrupts to be generated in loopback mode by writing the lower four bits of U0MCR.	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	-

12.6.8.1 Auto-flow control

If auto-RTS mode is enabled the UART‘s receiver FIFO hardware controls the RTS output of the UART. If the auto-CTS mode is enabled the UART‘s U0TSR hardware will only start transmitting if the CTS input signal is asserted.

12.6.8.1.1 Auto-RTS

The auto-RTS function is enabled by setting the RTSen bit. Auto-RTS data flow control originates in the U0RBR 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, RTS is de-asserted (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 de-assertion of RTS until after it has begun sending the additional byte. RTS is automatically reasserted (to a low value) once the receiver FIFO has reached the previous trigger level. The reassertion of RTS signals the sending UART to continue transmitting data.

If Auto-RTS mode is disabled, the RTSen bit controls the RTS output of the UART. If Auto-RTS mode is enabled, hardware controls the RTS output, and the actual value of 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 U0FCR set to 0x2, then, if Auto-RTS is enabled, the UART will de-assert the RTS output as soon as the receive FIFO contains 8 bytes (Table 201 on page 190). The RTS output will be reasserted as soon as the receive FIFO hits the previous trigger level: 4 bytes.

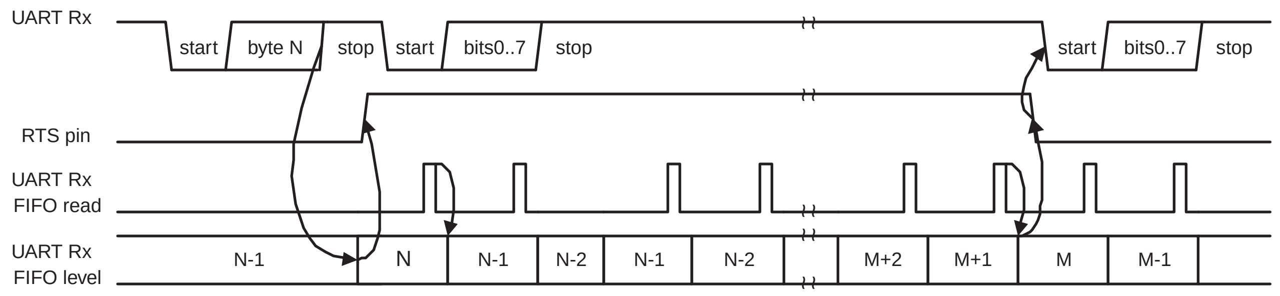


Fig 20. Auto-RTS Functional Timing

12.6.8.1.2 Auto-CTS

The Auto-CTS function is enabled by setting the CTSen bit. If Auto-CTS is enabled, the transmitter circuitry in the U0TSR module checks CTS input before sending the next data byte. When CTS is active (low), the transmitter sends the next byte. To stop the transmitter from sending the following byte, 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 CTS signal does not trigger a modem status interrupt unless the CTS Interrupt Enable bit is set, Delta CTS bit in the U0MSR will be set though. Table 204 lists the conditions for generating a Modem Status interrupt.

Table 204. Modem status interrupt generation

Enable modem status interrupt (U0ER[3])	CTSen (U0MCR[7])	CTS interrupt enable (U0IER[7])	Delta CTS (U0MSR[0])	Delta DCD or trailing edge RI or Delta DSR (U0MSR[3] or U0MSR[2] or U0MSR[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 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 21 illustrates the Auto-CTS functional timing.

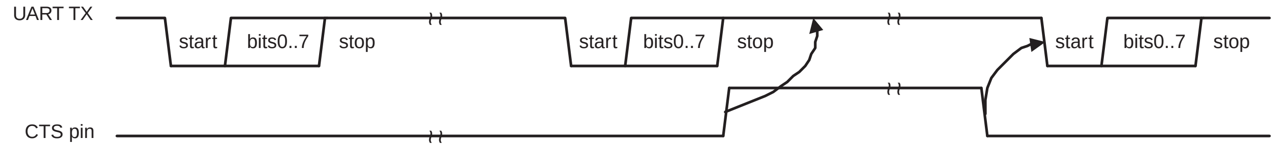


Fig 21. Auto-CTS Functional Timing

While starting transmission of the initial character, the $\overline{\text{CTS}}$ 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 $\overline{\text{CTS}}$ is de-asserted (high). As soon as $\overline{\text{CTS}}$ gets de-asserted, transmission resumes and a start bit is sent followed by the data bits of the next character.

12.6.9 UART Line Status Register (U0LSR - 0x4000 8014, Read Only)

The U0LSR is a Read Only register that provides status information on the UART TX and RX blocks.

Table 205. UART Line Status Register (U0LSR - address 0x4000 8014, Read Only) bit description

Bit	Symbol	Value	Description	Reset Value
0	RDR		Receiver Data Ready. U0LSR[0] is set when the U0RBR holds an unread character and is cleared when the UART RBR FIFO is empty.	0
		0	U0RBR is empty.	
		1	U0RBR contains valid data.	
1	OE		Overrun Error. The overrun error condition is set as soon as it occurs. A U0LSR read clears U0LSR[1]. U0LSR[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 U0LSR read clears U0LSR[2]. Time of parity error detection is dependent on U0FCR[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 205. UART Line Status Register (U0LSR - 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 U0LSR read clears U0LSR[3]. The time of the framing error detection is dependent on U0FCR0. 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 U0LSR read clears this status bit. The time of break detection is dependent on U0FCR[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 U0THR write.	1
		0	U0THR contains valid data.	
		1	U0THR is empty.	
6	TEMT		Transmitter Empty. TEMT is set when both U0THR and U0TSR are empty; TEMT is cleared when either the U0TSR or the U0THR contain valid data.	1
		0	U0THR and/or the U0TSR contains valid data.	
		1	U0THR and the U0TSR are empty.	
7	RXFE		Error in RX FIFO. U0LSR[7] is set when a character with a RX error such as framing error, parity error or break interrupt, is loaded into the U0RBR. This bit is cleared when the U0LSR register is read and there are no subsequent errors in the UART FIFO.	0
		0	U0RBR contains no UART RX errors or U0FCR[0]=0.	
		1	UART RBR contains at least one UART RX error.	
31:8	-	-	Reserved	-

12.6.10 UART Modem Status Register

The U0MSR is a read-only register that provides status information on the modem input signals. U0MSR[3:0] is cleared on U0MSR read. Note that modem signals have no direct effect on the UART operation. They facilitate the software implementation of modem signal operations.

Table 206. UART Modem Status Register (U0MSR - address 0x4000 8018) bit description

Bit	Symbol	Value	Description	Reset Value
0	DELTACTS		Set upon state change of input $\overline{\text{CTS}}$. Cleared on a U0MSR read.	0
		0	No change detected on modem input $\overline{\text{CTS}}$.	
		1	State change detected on modem input $\overline{\text{CTS}}$.	
1	DELTADSR		Set upon state change of input $\overline{\text{DSR}}$. Cleared on a U0MSR read.	0
		0	No change detected on modem input $\overline{\text{DSR}}$.	
		1	State change detected on modem input $\overline{\text{DSR}}$.	
2	TERI		Trailing Edge RI. Set upon low to high transition of input $\overline{\text{RI}}$. Cleared on a U0MSR read.	0
		0	No change detected on modem input, $\overline{\text{RI}}$.	
		1	Low-to-high transition detected on $\overline{\text{RI}}$.	
3	DELTADCD		Set upon state change of input $\overline{\text{DCD}}$. Cleared on a U0MSR read.	0
		0	No change detected on modem input $\overline{\text{DCD}}$.	
		1	State change detected on modem input $\overline{\text{DCD}}$.	
4	CTS		Clear To Send State. Complement of input signal $\overline{\text{CTS}}$. This bit is connected to U0MCR[1] in modem loopback mode.	0
5	DSR		Data Set Ready State. Complement of input signal $\overline{\text{DSR}}$. This bit is connected to U0MCR[0] in modem loopback mode.	0
6	RI		Ring Indicator State. Complement of input $\overline{\text{RI}}$. This bit is connected to U0MCR[2] in modem loopback mode.	0
7	DCD		Data Carrier Detect State. Complement of input $\overline{\text{DCD}}$. This bit is connected to U0MCR[3] in modem loopback mode.	0
31:8	-	-	Reserved	-

12.6.11 UART Scratch Pad Register (U0SCR - 0x4000 801C)

The U0SCR 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 U0SCR has occurred.

Table 207. UART Scratch Pad Register (U0SCR - address 0x4000 801C) bit description

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

12.6.12 UART Auto-baud Control Register (U0ACR - 0x4000 8020)

The UART Auto-baud Control Register (U0ACR) 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 208. Auto-baud Control Register (U0ACR - address 0x4000 8020) 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		Auto restart	0
		0	No restart	
		1	Restart in case of time-out (counter restarts at next UART Rx falling edge)	0
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 U0IIR.	
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 U0IIR.	
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

12.6.13 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 U0DLM and U0DLL accordingly.

Auto-baud is started by setting the U0ACR Start bit. Auto-baud can be stopped by clearing the U0ACR 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 U0ACR 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 U0ACR 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 U0IIR ABTOInt interrupt will get set if the interrupt is enabled (U0IER ABToIntEn is set and the auto-baud rate measurement counter overflows).
- The U0IIR ABEOInt interrupt will get set if the interrupt is enabled (U0IER ABEOIntEn is set and the auto-baud has completed successfully).

The auto-baud interrupts have to be cleared by setting the corresponding U0ACR ABTOIntClr and ABEOIntEn bits.

The fractional baud rate generator must be disabled (DIVADDVAL = 0) during auto-baud. Also, when auto-baud is used, any write to U0DLM and U0DLL registers should be done before U0ACR register write. The minimum and the maximum baud rates supported by UART are function of UART_PCLK, number of data bits, stop bits and parity bits.

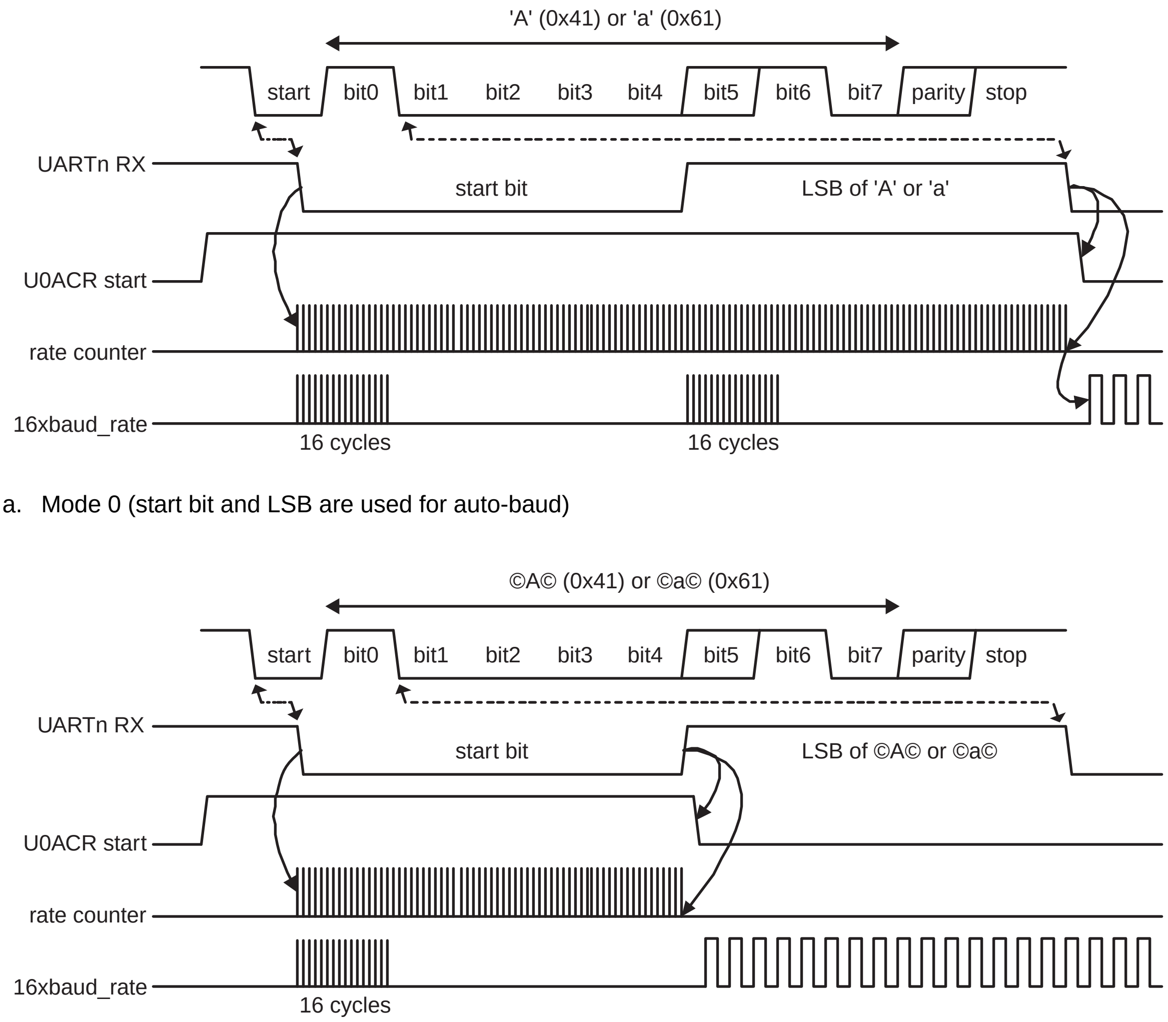
$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax \tag{2}$$

12.6.14 Auto-baud modes

When the software is expecting an ”AT" command, it configures the UART with the expected character format and sets the U0ACR Start bit. The initial values in the divisor latches U0DLM and U0DLM 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 U0ACR Start bit is set, the auto-baud protocol will execute the following phases:

1. On U0ACR Start bit setting, the baud rate measurement counter is reset and the UART U0RSR is reset. The U0RSR baud rate is switched 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.
3. During the receipt of the start bit, 16 pulses are generated on the RSR baud input with the frequency of the UART input clock, guaranteeing the start bit is stored in the U0RSR.

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, the rate counter will stop on next falling edge of the UART Rx pin. If Mode = 1, the rate counter will stop on the next rising edge of the UART Rx pin.
6. The rate counter is loaded into U0DLM/U0DLL and the baud rate will be switched to normal operation. After setting the U0DLM/U0DLL, the end of auto-baud interrupt U0IIR ABEOInt will be set, if enabled. The U0RSR 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 22. Auto-baud a) mode 0 and b) mode 1 waveform

12.6.15 UART Fractional Divider Register (U0FDR - 0x4000 8028)

The UART Fractional Divider Register (U0FDR) 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 209. UART Fractional Divider Register (U0FDR - 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 UART baud rate.	0
7:4	MULVAL	Baud rate pre-scaler multiplier value. This field must be greater or equal 1 for UART 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 baud rate can be calculated as:

$$UART_{baudrate} = \frac{PCLK}{16 \times (256 \times U0DLM + U0DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)} \tag{3}$$

Where UART_PCLK is the peripheral clock, U0DLM and U0DLL are the standard UART baud rate divider registers, and DIVADDVAL and MULVAL are UART fractional baud rate generator specific parameters.

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

1. 1 ≤ MULVAL ≤ 15
2. 0 ≤ DIVADDVAL ≤ 14
3. DIVADDVAL< MULVAL

The value of the U0FDR should not be modified while transmitting/receiving data or data may be lost or corrupted.

If the U0FDR 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.

12.6.15.1 Baud rate calculation

UART can operate with or without using the Fractional Divider. In real-life applications it is likely that the desired baud rate 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 baud rate with a relative error of less than 1.1% from the desired one.

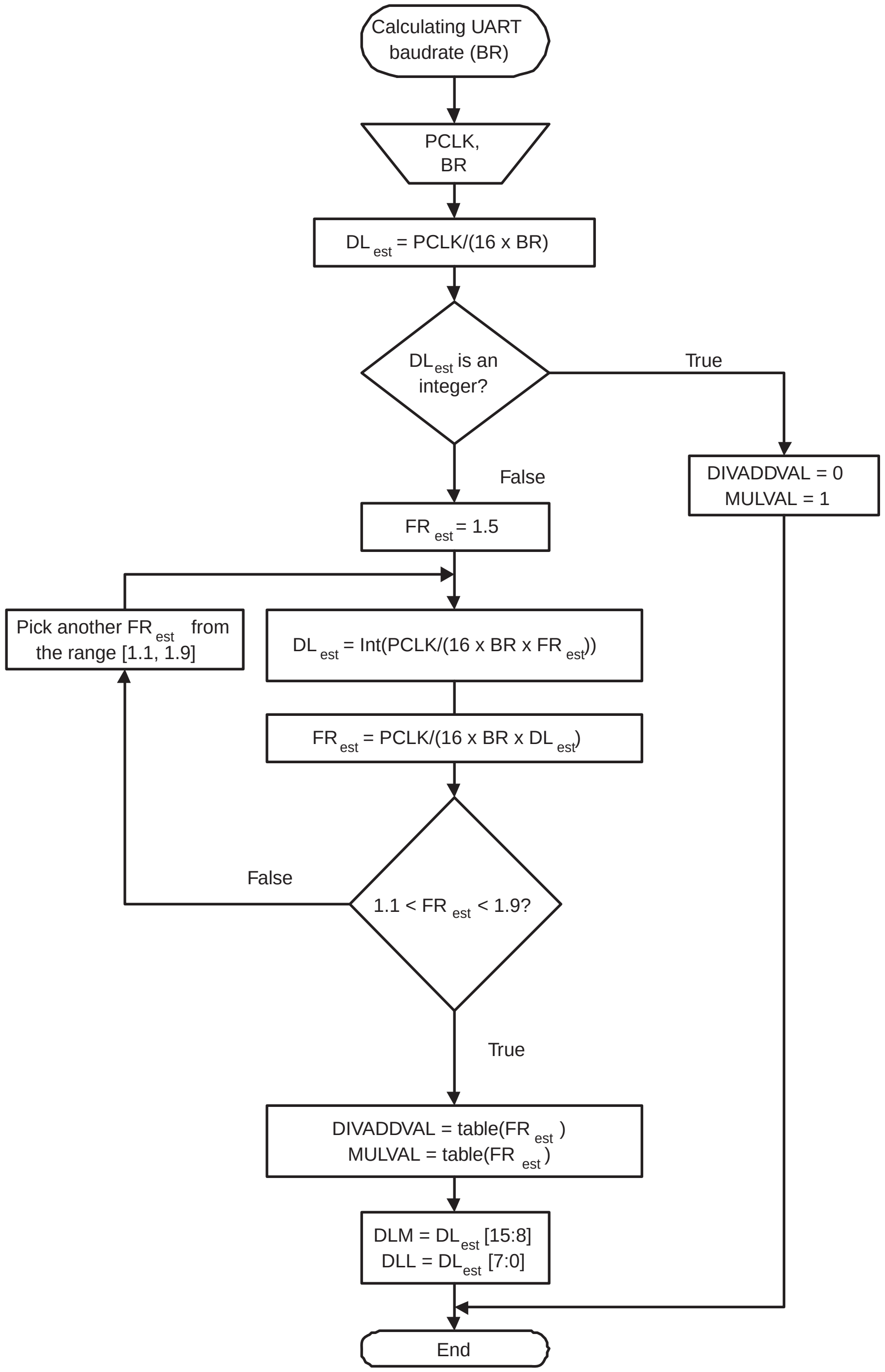

Calculating UART baudrate (BR)
PCLK, BR
DL_{est} = PCLK/(16 x BR)
DL_{est} is an integer?
True
DIVADDVAL = 0
MULVAL = 1
False
FR_{est} = 1.5
Pick another FR_{est} from the range [1.1, 1.9]
DL_{est} = Int(PCLK/(16 x BR x FR_{est}))
FR_{est} = PCLK/(16 x BR x DL_{est})
False
1.1 < FR_{est} < 1.9?
True
DIVADDVAL = table(FR_{est})
MULVAL = table(FR_{est})
DLM = DL_{est} [15:8]
DLL = DL_{est} [7:0]
End

Fig 23. Algorithm for setting UART dividers

Table 210. 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

12.6.15.1.1 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.

12.6.15.1.2 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 210 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, the UART's baud rate is 115384. This rate has a relative error of 0.16% from the originally specified 115200.

12.6.16 UART Transmit Enable Register (U0TER - 0x4000 8030)

In addition to being equipped with full hardware flow control (auto-cts and auto-rts mechanisms described above), U0TER enables implementation of software 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.

All information provided in this document is subject to legal disclaimers.

Although Table 211 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.

Table 211 describes how to use TXEn bit in order to achieve software flow control.

Table 211. UART Transmit Enable Register (U0TER - 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 ($\overline{\text{CTS}}$) 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	-

12.6.17 UART RS485 Control register (U0RS485CTRL - 0x4000 804C)

The U0RS485CTRL register controls the configuration of the UART in RS-485/EIA-485 mode.

Table 212. UART RS485 Control register (U0RS485CTRL - 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.	
3	SEL		Direction control pins select	0
		0	If direction control is enabled (bit DCTRL = 1), pin $\overline{\text{RTS}}$ is used for direction control.	
		1	If direction control is enabled (bit DCTRL = 1), pin $\overline{\text{DTR}}$ is used for direction control.	

Table 212. UART RS485 Control register (U0RS485CTRL - address 0x4000 804C) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
4	DCTRL		Direction control enable	0
		0	Disable Auto Direction Control.	
		1	Enable Auto Direction Control.	
5	OINV		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

12.6.18 UART RS485 Address Match register (U0RS485ADRMATCH - 0x4000 8050)

The U0RS485ADRMATCH register contains the address match value for RS-485/EIA-485 mode.

Table 213. UART RS-485 Address Match register (U0RS485ADRMATCH - address 0x4000 8050) bit description

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

12.6.19 UART1 RS485 Delay value register (U0RS485DLY - 0x4000 8054)

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 214. UART RS-485 Delay value register (U0RS485DLY - 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

12.6.20 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.

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.

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 an 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.

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 $\overline{\text{RTS}}$ pin when RS485CTRL bit 3 = ‘0’. It will use the $\overline{\text{DTR}}$ 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.

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 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.

RS485/EIA-485 output inversion

The polarity of the direction control signal on the RTS (or DTR) pins can be reversed by programming bit 5 in the U0RS485CTRL 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.

12.7 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, U0RX, monitors the serial input line, RXD, for valid input. The UART RX Shift Register (U0RSR) accepts valid characters via RXD. After a valid character is assembled in the U0RSR, 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, U0TX, accepts data written by the CPU or host and buffers the data in the UART TX Holding Register FIFO (U0THR). The UART TX Shift Register (U0TSR) reads the data stored in the U0THR and assembles the data to transmit via the serial output pin, TXD1.

The UART Baud Rate Generator block, U0BRG, generates the timing enables used by the UART TX block. The U0BRG clock input source is UART_PCLK. The main clock is divided down per the divisor specified in the U0DLL and U0DLM registers. This divided down clock is a 16x oversample clock, NBAUDOUT.

The interrupt interface contains registers U0IER and U0IIR. The interrupt interface receives several one clock wide enables from the U0TX and U0RX blocks.

Status information from the U0TX and U0RX is stored in the U0LSR. Control information for the U0TX and U0RX is stored in the U0LCR.

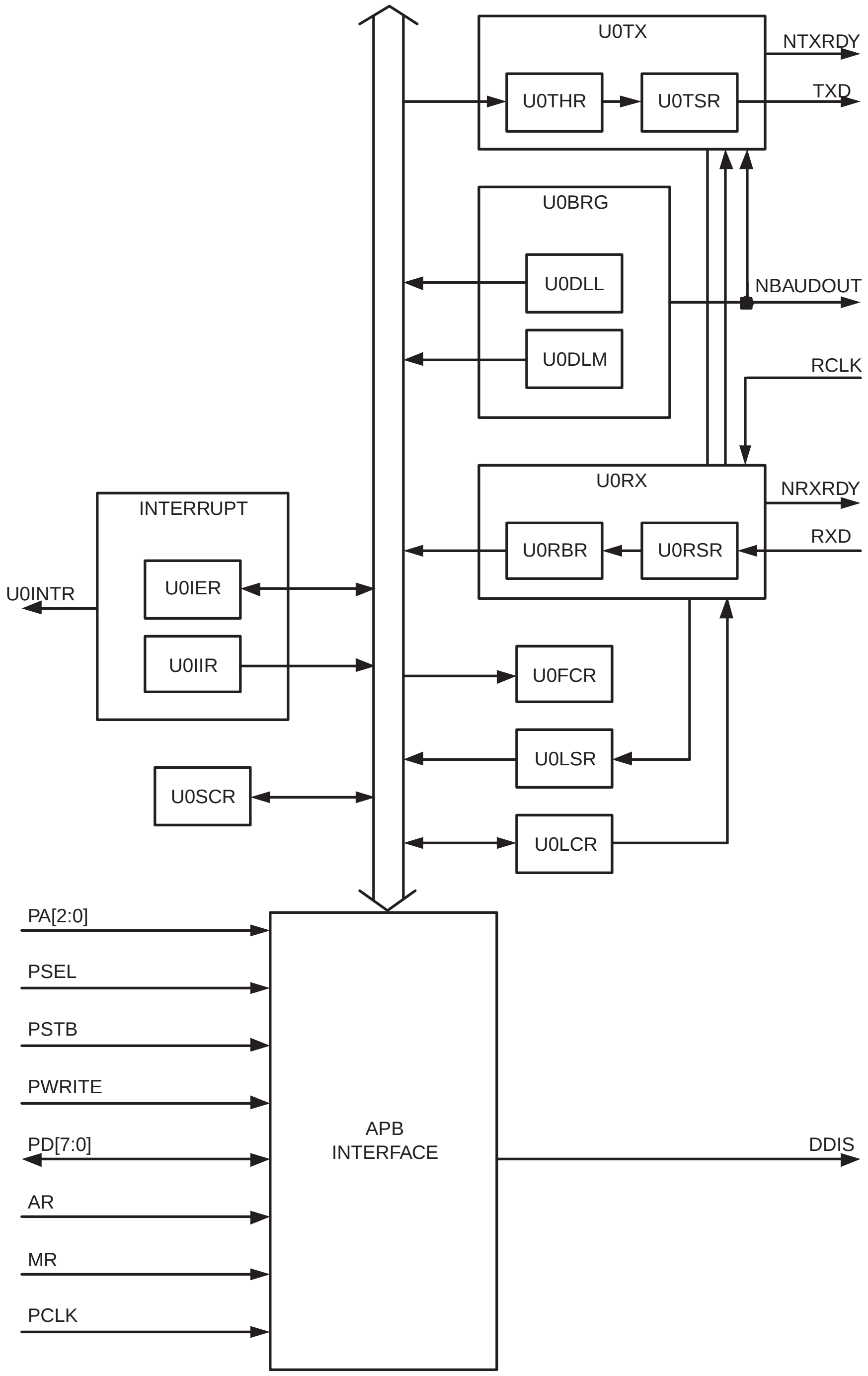


Fig 24. UART block diagram

UM10375

Chapter 13: LPC13xx I2C-bus controller

Rev. 5 — 21 June 2012 **User manual**

13.1 How to read this chapter

The I^{2}C-bus block is identical for all LPC13xx parts.

13.2 Basic configuration

The I^{2}C-bus interface is configured using the following registers:

1. Pins: The I2C pin functions and the I2C mode are configured in the IOCONFIG register block (Table 107 and Table 108).
2. Power and peripheral clock: In the SYSAHBCLKCTRL register, set bit 5 (Table 25).
3. Reset: Before accessing the I2C block, ensure that the I2C_RST_N bit (bit 1) in the PRESETCTRL register (Table 9) is set to 1. This de-asserts the reset signal to the I2C block.

13.3 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-bus contains a standard I^{2}C-compliant bus interface with two pins.

13.4 Applications

Interfaces to external I^{2}C standard parts, such as serial RAMs, LCDs, tone generators, other microcontrollers, etc.

13.5 General description

A typical I^{2}C-bus configuration is shown in Figure 25. 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^{2}C bus will not be released.

The I^{2}C interface is byte oriented and has four operating modes: master transmitter mode, master receiver mode, slave transmitter mode and slave receiver mode.

The I^{2}C interface complies with the entire I^{2}C specification, supporting the ability to turn power off to the ARM Cortex-M3 without interfering with other devices on the same I^{2}C-bus.

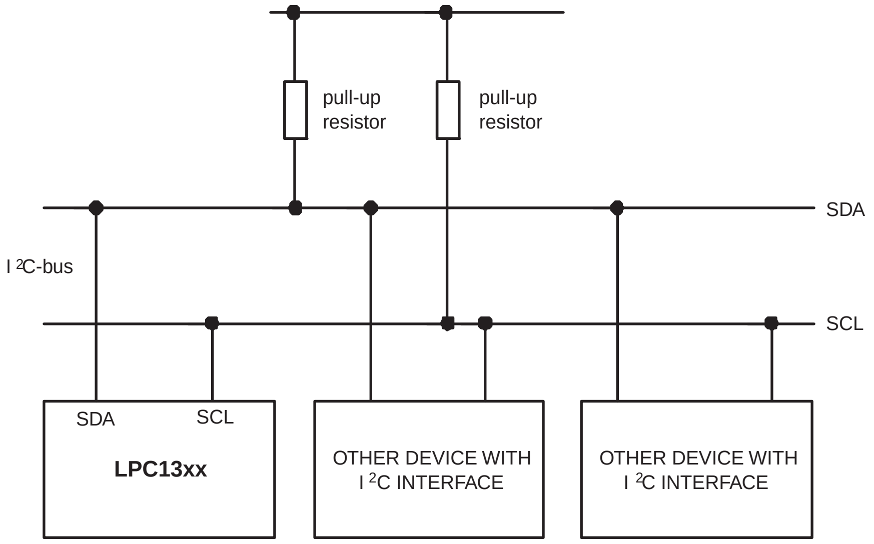


Fig 25. I^{2}C-bus configuration

13.5.1 I^{2}C Fast-mode Plus

Fast-Mode Plus supports a 1 Mbit/sec transfer rate to communicate with the I^{2}C-bus products which NXP Semiconductors is now providing.

In order to use Fast-Mode Plus, the I^{2}C pins must be properly configured in the IOCONFIG register block, see Table 107 and Table 108. In Fast-mode Plus, rates above 400 kHz and up to 1 MHz may be selected.

13.6 Pin description

Table 215. I²C-bus pin description

Pin	Type	Description
SDA	Input/Output	I²C-bus Serial Data
SCL	Input/Output	I²C-bus Serial Clock

The I²C-bus pins must be configured through the IOCON_PIO0_4 (Table 107) and IOCON_PIO0_5 (Table 108) registers for Standard/ Fast-mode or Fast-mode Plus. In these modes, the I²C-bus pins are open-drain outputs and fully compatible with the I²C-bus specification.

13.7 Clocking and power control

The clock to the I²C-bus interface (PCLK_I2C) is provided by the system clock (see Figure 3). This clock can be disabled through bit 5 in the SYSAHBCLKCTRL register (Table 25) for power savings.

Remark: Before accessing the I2C block, ensure that the I2C_RST_N bit (bit 1) in the PRESETCTRL register (Table 9) is set to 1. This de-asserts the reset signal to the I2C block.

13.8 Register description

Table 216. Register overview: I²C (base address 0x4000 0000)

Name	Access	Address offset	Description	Reset value[1]
I2C0CONSET	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
I2C0STAT	RO	0x004	**I2C Status Register.** During I²C operation, this register provides detailed status codes that allow software to determine the next action needed.	0xF8
I2C0DAT	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
I2C0ADR0	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
I2C0SCLH	R/W	0x010	**SCH Duty Cycle Register High Half Word.** Determines the high time of the I²C clock.	0x04
I2C0SCLL	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
I2C0CONCLR	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
I2C0MMCTRL	R/W	0x01C	**Monitor mode control register.**	0x00

Table 216. Register overview: I^{2}C (base address 0x4000 0000) *…continued*

Name	Access	Address offset	Description	Reset value[1]
I2C0ADR1	R/W	0x020	**I2C Slave Address Register 1.** Contains the 7-bit slave address for operation of the I^{2}C 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
I2C0ADR2	R/W	0x024	**I2C Slave Address Register 2.** Contains the 7-bit slave address for operation of the I^{2}C 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
I2C0ADR3	R/W	0x028	**I2C Slave Address Register 3.** Contains the 7-bit slave address for operation of the I^{2}C 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
I2C0DATA_ 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
I2C0MASK0	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
I2C0MASK1	R/W	0x034	**I2C Slave address mask register 1**. This mask register is associated with I2ADR1 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	0x00
I2C0MASK2	R/W	0x038	**I2C Slave address mask register 2**. This mask register is associated with I2ADR2 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	0x00
I2C0MASK3	R/W	0x03C	**I2C Slave address mask register 3**. This mask register is associated with I2ADR3 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.

13.8.1 I^{2}C Control Set register (I2C0CONSET - 0x4000 0000)

The I2CONSET registers control setting of bits in the I2CON register that controls operation of the I^{2}C interface. Writing a one to a bit of this register causes the corresponding bit in the I^{2}C control register to be set. Writing a zero has no effect.

Table 217. I^{2}C Control Set register (I2C0CONSET - 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	I^{2}C interrupt flag.	0
4	STO	STOP flag.	0
5	STA	START flag.	0
6	I2EN	I^{2}C interface enable.	0
31:7	-	Reserved. The value read from a reserved bit is not defined.	-

I2EN I^{2}C Interface Enable. When I2EN is 1, the I^{2}C interface is enabled. I2EN can be cleared by writing 1 to the I2ENC bit in the I2CONCLR register. When I2EN is 0, the I^{2}C interface is disabled.

When I2EN is “0”, the SDA and SCL input signals are ignored, the I^{2}C 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^{2}C-bus since, when I2EN is reset, the I^{2}C-bus status is lost. The AA flag should be used instead.

STA is the START flag. Setting this bit causes the I^{2}C 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^{2}C 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^{2}C 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^{2}C 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^{2}C-bus if it the interface is in master mode, and transmits a START condition thereafter. If the I^{2}C 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^{2}C 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^{2}C-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^{2}C Interrupt Flag. This bit is set when the I^{2}C 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^{2}C is in the master receiver mode.
4. A data byte has been received while the I^{2}C 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^{2}C is in the master receiver mode.
2. A data byte has been received while the I^{2}C is in the addressed slave receiver mode.

13.8.2 I^{2}C Status register (I2C0STAT - 0x4000 0004)

Each I^{2}C Status register reflects the condition of the corresponding I^{2}C interface. The I^{2}C Status register is Read-Only.

Table 218. I^{2}C Status register (I2C0STAT - 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^{2}C interface.	0x1F
31:8	-	Reserved. The value read from a reserved bit is not defined.	-

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^{2}C 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 235 to Table 238.

13.8.3 I^{2}C Data register (I2C0DAT - 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 219. I^{2}C Data register (I2C0DAT - 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. The value read from a reserved bit is not defined.	-

13.8.4 I^{2}C Slave Address register 0 (I2C0ADR0- 0x4000 000C)

This register is readable and writable and are only used when an I^{2}C 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. The slave address register will be cleared to this disabled state on reset. See also Table 226.

Table 220. I^{2}C Slave Address register 0 (I2C0ADR0- 0x4000 000C) bit description

Bit	Symbol	Description	Reset value
0	GC	General Call enable bit.	0
7:1	Address	The I^{2}C device address for slave mode.	0x00
31:8	-	Reserved. The value read from a reserved bit is not defined.	-

13.8.5 I^{2}C SCL HIGH and LOW duty cycle registers (I2C0SCLH - 0x4000 0010 and I2C0SCLL- 0x4000 0014)

Table 221. I^{2}C SCL HIGH Duty Cycle register (I2C0SCLH - address 0x4000 0010) bit description

Bit	Symbol	Description	Reset value
15:0	SCLH	Count for SCL HIGH time period selection.	0x0004
31:16	-	Reserved. The value read from a reserved bit is not defined.	-

Table 222. I^{2}C SCL Low duty cycle register (I2C0SCLL - 0x4000 0014) bit description

Bit	Symbol	Description	Reset value
15:0	SCLL	Count for SCL low time period selection.	0x0004
31:16	-	Reserved. The value read from a reserved bit is not defined.	-

13.8.5.1 Selecting the appropriate I^{2}C 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 I2C clock):

$$I^2C_{bitfrequency} = \frac{I2CPCLK}{I2CSCLH + I2CSCLL} \tag{4}$$

The values for I2SCLL and I2SCLH must ensure that the data rate is in the appropriate I^{2}C data rate range. Each register value must be greater than or equal to 4. Table 223 gives some examples of I^{2}C-bus rates based on I2C_PCLK frequency and I2SCLL and I2SCLH values.

Table 223. I2SCLL + I2SCLH values for selected I^{2}C clock values

I^{2}C mode	I^{2}C bit frequency	PCLK_I2C (MHz)										
		6	8	10	12	16	20	30	40	50	60	70
		I2SCLH + I2SCLL										
Standard mode	100 kHz	60	80	100	120	160	200	300	400	500	600	700
Fast-mode	400 kHz	15	20	25	30	40	50	75	100	125	150	175
Fast-mode Plus	1 MHz	-	8	10	12	16	20	30	40	50	60	70

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^{2}C-bus specification defines the SCL low time and high time at different values for a Fast-mode and Fast-mode Plus I^{2}C.

13.8.6 I^{2}C Control Clear register (I2C0CONCLR - 0x4000 0018)

The I2CONCLR registers control clearing of bits in the I2CON register that controls operation of the I^{2}C interface. Writing a one to a bit of this register causes the corresponding bit in the I^{2}C control register to be cleared. Writing a zero has no effect.

Table 224. I^{2}C Control Clear register (I2C0CONCLR - 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^{2}C 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^{2}C 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. The value read from a reserved bit is not defined.	-

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^{2}C 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^{2}C Interface Disable bit. Writing a 1 to this bit clears the I2EN bit in the I2CONSET register. Writing 0 has no effect.

13.8.7 I^{2}C Monitor mode control register (I2C0MMCTRL - 0x4000 001C)

This register controls the Monitor mode which allows the I^{2}C module to monitor traffic on the I^{2}C bus without actually participating in traffic or interfering with the I^{2}C bus.

Table 225. I^{2}C Monitor mode control register (I2C0MMCTRL - 0x4000 001C) bit description

Bit	Symbol	Value	Description	Reset value
0	MM_ENA		Monitor mode enable.	0
		0	Monitor mode disabled.	
		1	The I^{2}C module will enter monitor mode. In this mode the SDA output will be forced high. This will prevent the I^{2}C module from outputting data of any kind (including ACK) onto the I^{2}C 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 I^{2}C 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 I^{2}C clock line.	
		1	When this bit is set, the I^{2}C 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 I^{2}C module can “stretch” the clock line (hold it low) until it has had time to respond to an I^{2}C interrupt.[1]	
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 I^{2}C 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. The value read from reserved bits is not defined.	

[1] When the ENA_SCL bit is cleared and the I^{2}C no longer has the ability to stall the bus, interrupt response time becomes important. To give the part more time to respond to an I^{2}C interrupt under these conditions, a DATA _BUFFER register is used (Section 13.8.9) to hold received data for a full 9-bit word transmission time.

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).

13.8.7.1 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.

13.8.7.2 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.

13.8.8 I²C Slave Address registers (I2C0ADR[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 will be cleared to this disabled state on reset.

Table 226. I²C Slave Address registers (I2C0ADR[1, 2, 3]- 0x4000 00[20, 24, 28]) 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. The value read from a reserved bit is not defined.	0

13.8.9 I²C Data buffer register (I2C0DATA_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 227. I^{2}C Data buffer register (I2C0DATA_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. The value read from a reserved bit is not defined.	0

13.8.10 I^{2}C Mask registers (I2C0MASK[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 (I2DAT) to determine what the received address was that actually caused the match.

Table 228. I^{2}C Mask registers (I2C0MASK[0, 1, 2, 3] - 0x4000 00[30, 34, 38, 3C]) 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. The value read from reserved bits is undefined.	0

13.9 I^{2}C operating modes

In a given application, the I^{2}C block may operate as a master, a slave, or both. In the slave mode, the I^{2}C 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 I^{2}C block switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

13.9.1 Master Transmitter mode

In this mode data is transmitted from master to slave. Before the master transmitter mode can be entered, the I2CONSET register must be initialized as shown in Table 229. I2EN must be set to 1 to enable the I^{2}C function. If the AA bit is 0, the I^{2}C 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 I2CONCLR register. THe STA bit should be cleared after writing the slave address.

Table 229. I2C0CONSET and I2C1CONSET 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^{2}C interface will enter master transmitter mode when software sets the STA bit. The I^{2}C 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 I2STAT 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 I2DAT register, and then clear the SI bit. SI is cleared by writing a 1 to the SIC bit in the I2CONCLR 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 235 to Table 238.

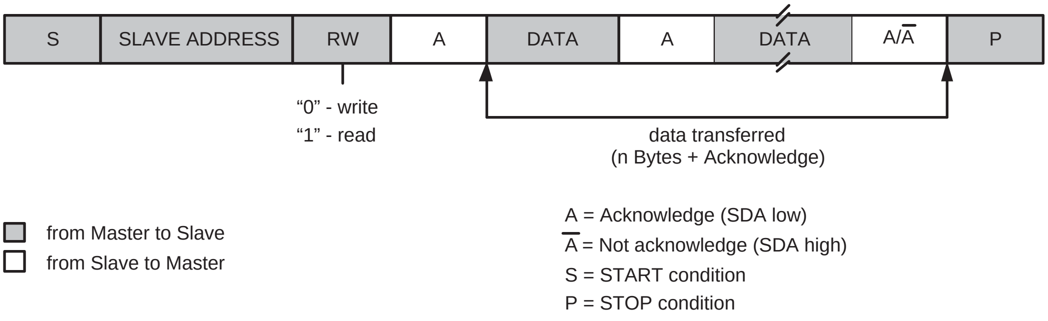


Fig 26. Format in the Master Transmitter mode

13.9.2 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^{2}C Data register (I2DAT), 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 236.

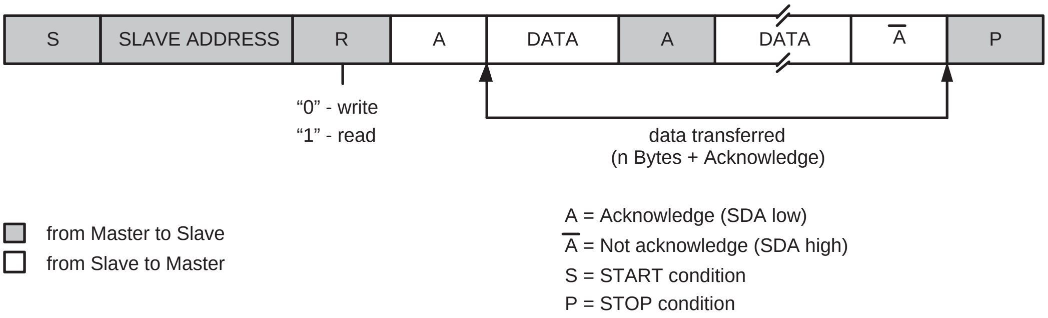


Fig 27. Format of Master Receiver mode

After a Repeated START condition, I^{2}C may switch to the master transmitter mode.

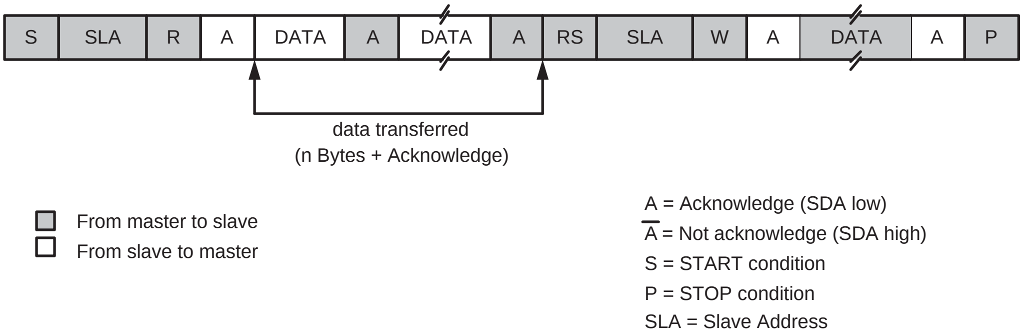


Fig 28. A Master Receiver switches to Master Transmitter after sending Repeated START

13.9.3 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 (I2ADR0-3) and write the I^{2}C Control Set register (I2CONSET) as shown in Table 230.

Table 230. I2C0CONSET and I2C1CONSET 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^{2}C function. AA bit must be set to 1 to acknowledge its own slave address or the General Call address. The STA, STO and SI bits are set to 0.

After I2ADR and I2CONSET are initialized, the I^{2}C interface waits until it is addressed by its own address or general 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 (I2STAT). Refer to Table 237 for the status codes and actions.

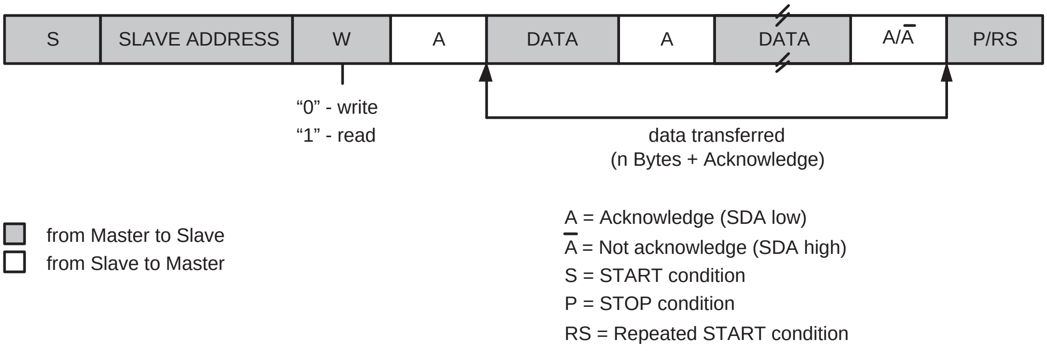


Fig 29. Format of Slave Receiver mode

13.9.4 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^{2}C may operate as a master and as a slave. In the slave mode, the I^{2}C hardware looks for its own slave address 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^{2}C interface switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

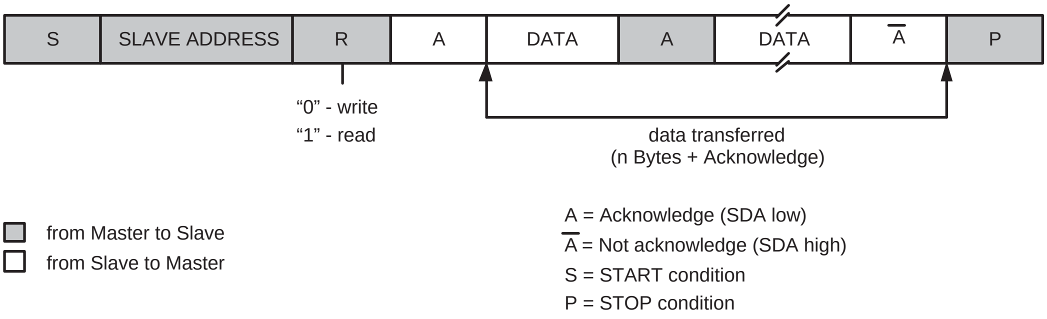


Fig 30. Format of Slave Transmitter mode

13.10 I^{2}C implementation and operation

Figure 31 shows how the on-chip I^{2}C-bus interface is implemented, and the following text describes the individual blocks.

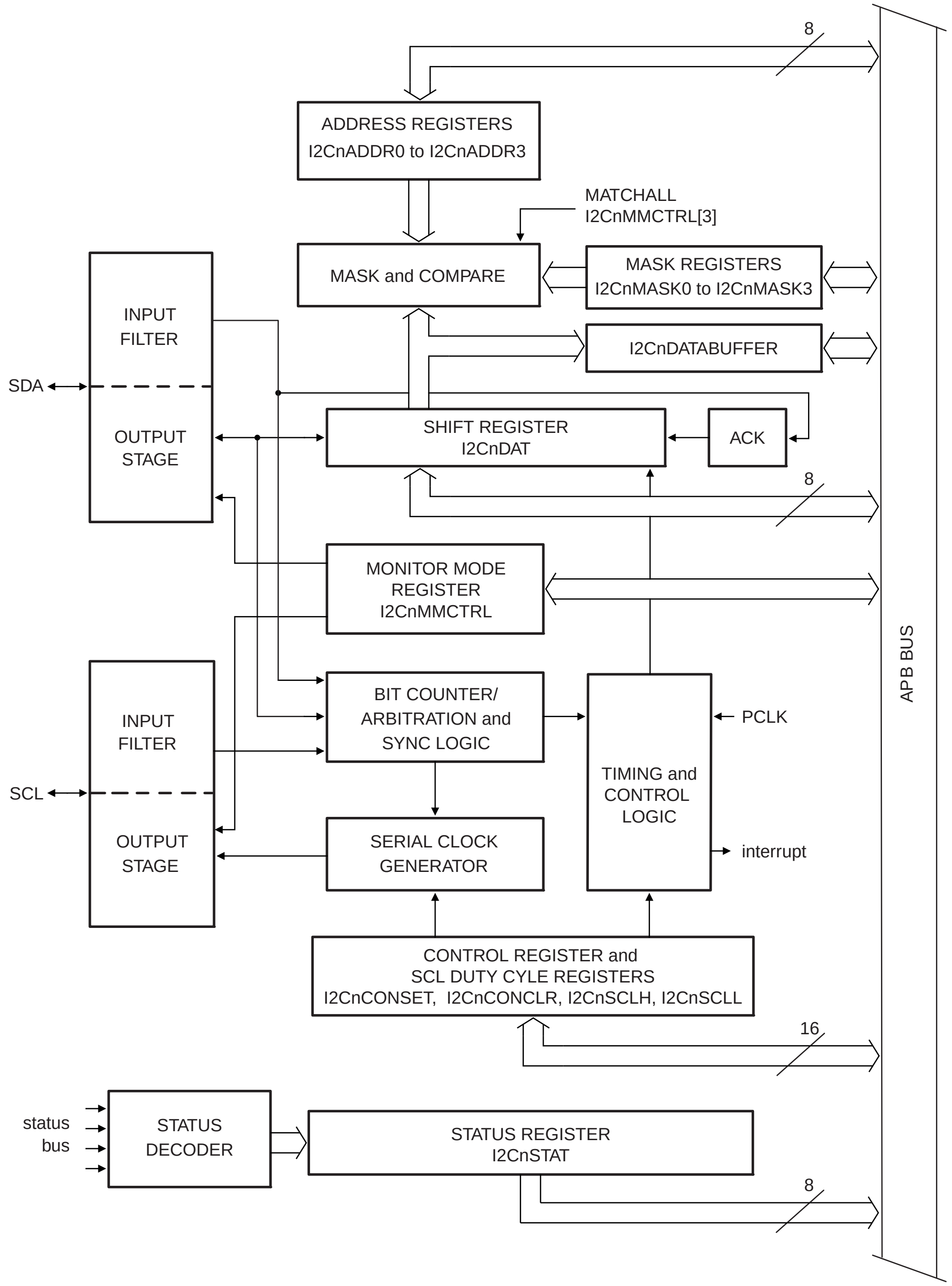


Fig 31. I^{2}C serial interface block diagram

13.10.1 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 I^{2}C is a special pad designed to conform to the I^{2}C specification.

13.10.2 Address Registers, I2ADDR0 to I2ADDR3

These registers may be loaded with the 7-bit slave address (7 most significant bits) to which the I^{2}C 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 I2DAT register at the state where the own slave address has been received.

13.10.3 Address mask registers, I2MASK0 to I2MASK3

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.

If the I2ADDRn bit 0 (GC enable bit) is as set and bits(7:1) are all zeroes, then the part will respond to a received address = “0000000” regardless of the state of the associated mask register.

When an address-match interrupt occurs, the processor will have to read the data register (I2DAT) to determine what the received address was that actually caused the match.

13.10.4 Comparator

The comparator compares the received 7-bit slave address with its own slave address (7 most significant bits in I2ADR). It also compares the first received 8-bit byte with the General Call address (0x00). If an equality is found, the appropriate status bits are set and an interrupt is requested.

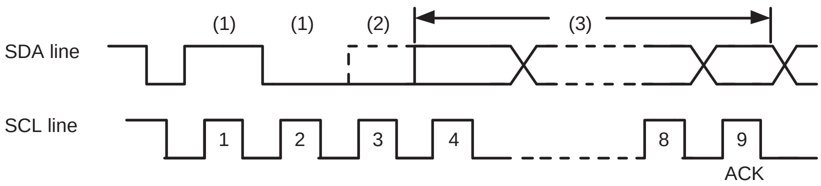
13.10.5 Shift register, I2DAT

This 8-bit register contains a byte of serial data to be transmitted or a byte which has just been received. 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. While data is being shifted out, data on the bus is simultaneously being shifted in; I2DAT 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 I2DAT.

13.10.6 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^{2}C-bus. If another device on the bus overrules a logic 1 and pulls the SDA line low, arbitration is lost, and the I^{2}C block immediately changes from master transmitter to slave receiver. The I^{2}C 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^{2}C 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^{2}C block generates no further clock pulses. Figure 32 shows the arbitration procedure.



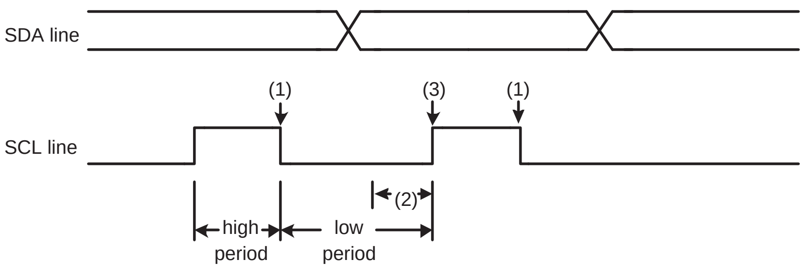
(1) Another device transmits serial data.

(2) Another device overrules a logic (dotted line) transmitted this I^{2}C master by pulling the SDA line low. Arbitration is lost, and this I^{2}C enters Slave Receiver mode.

(3) This I^{2}C is in Slave Receiver mode but still generates clock pulses until the current byte has been transmitted. This I^{2}C will not generate clock pulses for the next byte. Data on SDA originates from the new master once it has won arbitration.

Fig 32. 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 33 shows the synchronization procedure.



(1) Another device pulls the SCL line low before this I^{2}C 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 I^{2}C has timed a complete low time and released SCL. The I^{2}C 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 33. 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 I^{2}C 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.

13.10.7 Serial clock generator

This programmable clock pulse generator provides the SCL clock pulses when the I^{2}C block is in the master transmitter or master receiver mode. It is switched off when the I^{2}C block is in a slave mode. The I^{2}C output clock frequency and duty cycle is programmable via the I^{2}C Clock Control Registers. See the description of the I2CSCLL and I2CSCLH 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.

13.10.8 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 I2DAT, 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 I^{2}C-bus status.

13.10.9 Control register, I2CONSET and I2CONCLR

The I^{2}C control register contains bits used to control the following I^{2}C block functions: start and restart of a serial transfer, termination of a serial transfer, bit rate, address recognition, and acknowledgment.

The contents of the I^{2}C control register may be read as I2CONSET. Writing to I2CONSET will set bits in the I^{2}C control register that correspond to ones in the value written. Conversely, writing to I2CONCLR will clear bits in the I^{2}C control register that correspond to ones in the value written.

13.10.10 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 I^{2}C-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 I^{2}C 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).

13.11 Details of I^{2}C 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 34, Figure 35, Figure 36, Figure 37, and Figure 38. Table 231 lists abbreviations used in these figures when describing the I^{2}C operating modes.

Table 231. Abbreviations used to describe an I^{2}C 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

In Figure 34 to Figure 38, circles are used to indicate when the serial interrupt flag is set. The numbers in the circles show the status code held in the I2STAT 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 I2STAT 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 235 to Table 239.

13.11.1 Master Transmitter mode

In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see Figure 34). Before the master transmitter mode can be entered, I2CON must be initialized as follows:

Table 232. I2CONSET 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^{2}C rate must also be configured in the I2SCLL and I2SCLH registers. I2EN must be set to logic 1 to enable the I^{2}C block. If the AA bit is reset, the I^{2}C 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^{2}C 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^{2}C logic will now test the I^{2}C-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 (I2STAT) will be 0x08. This status code is used by the interrupt service routine to enter the appropriate state service routine that loads I2DAT with the slave address and the data direction bit (SLA+W). The SI bit in I2CON 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 I2STAT 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 235. After a Repeated START condition (state 0x10). The I^{2}C block may switch to the master receiver mode by loading I2DAT with SLA+R).

Table 233. Master Transmitter mode

Status Code (I2CSTAT)	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From I2DAT	To I2CON				
			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+R 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 I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT 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 I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x28	Data byte in I2DAT 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 I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x30	Data byte in I2DAT 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 I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT 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 I2DAT action or	0	0	0	X	I^2C-bus will be released; not addressed slave will be entered.
		No I2DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.

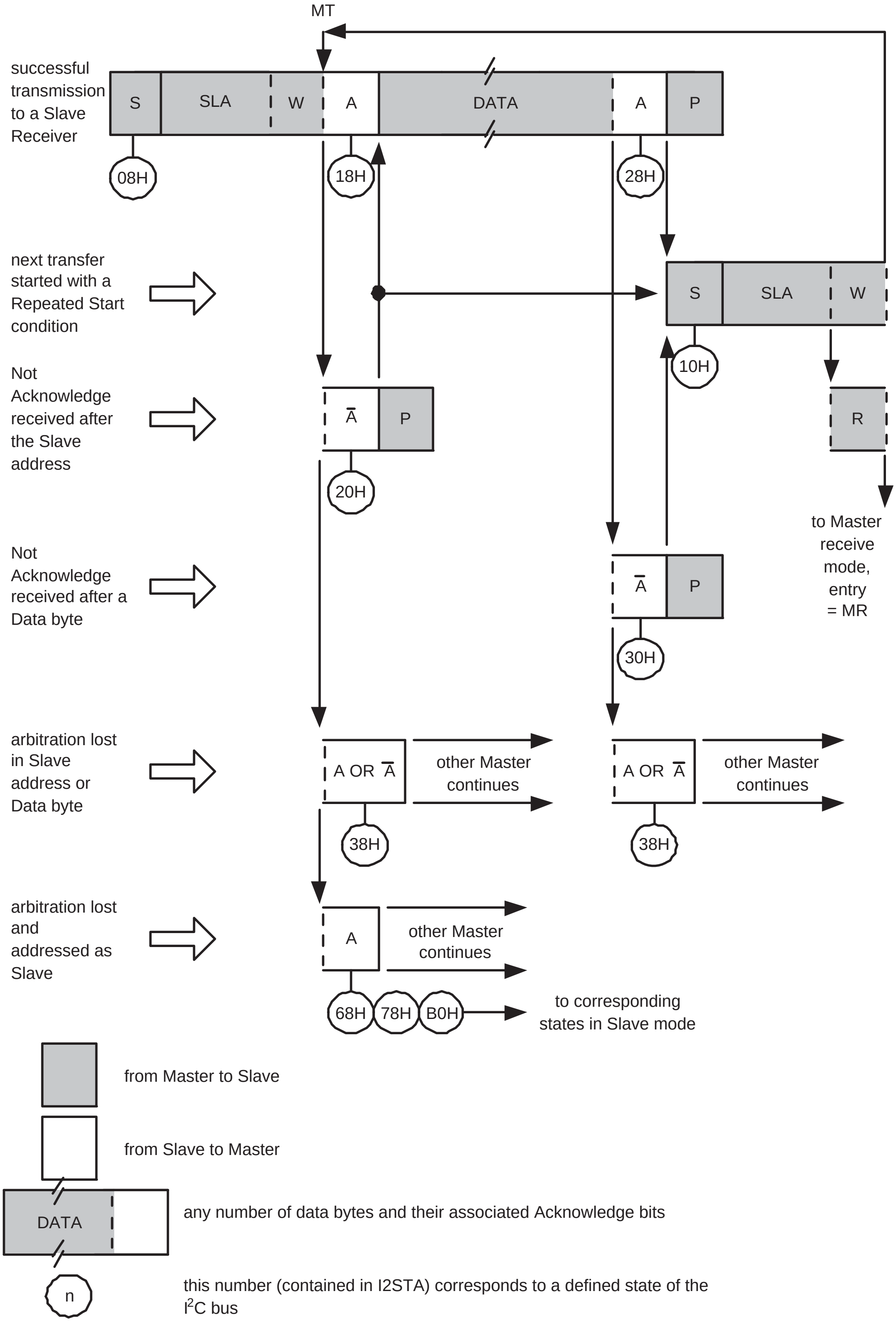

MT
successful transmission to a Slave Receiver
S
SLA
W
A
DATA
A
P
08H
18H
28H
next transfer started with a Repeated Start condition
S
SLA
W
10H
Not Acknowledge received after the Slave address
Ā
P
20H
R
to Master receive mode, entry = MR
Not Acknowledge received after a Data byte
Ā
P
30H
arbitration lost in Slave address or Data byte
A OR Ā
other Master continues
38H
A OR Ā
other Master continues
38H
arbitration lost and addressed as Slave
A
other Master continues
68H
78H
B0H
to corresponding states in Slave mode
from Master to Slave
from Slave to Master
DATA
any number of data bytes and their associated Acknowledge bits
n
this number (contained in I2STA) corresponds to a defined state of the I²C bus

Fig 34. Format and states in the Master Transmitter mode

13.11.2 Master Receiver mode

In the master receiver mode, a number of data bytes are received from a slave transmitter (see Figure 35). The transfer is initialized as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load I2DAT with the 7-bit slave address and the data direction bit (SLA+R). The SI bit in I2CON 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 I2STAT 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 236. After a Repeated START condition (state 0x10), the I^2C block may switch to the master transmitter mode by loading I2DAT with SLA+W.

Table 234. Master Receiver mode

Status Code (I2CSTAT)	Status of the I^{2}C-bus and hardware	Application software response					Next action taken by I^{2}C hardware
		To/From I2DAT	To I2CON				
			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^{2}C block will be switched to MST/TRX mode.
0x38	Arbitration lost in NOT ACK bit.	No I2DAT action or	0	0	0	X	I^{2}C-bus will be released; the I^{2}C block will enter a slave mode.
		No I2DAT 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 I2DAT action or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		No I2DAT 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 I2DAT action or	1	0	0	X	Repeated START condition will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT 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.

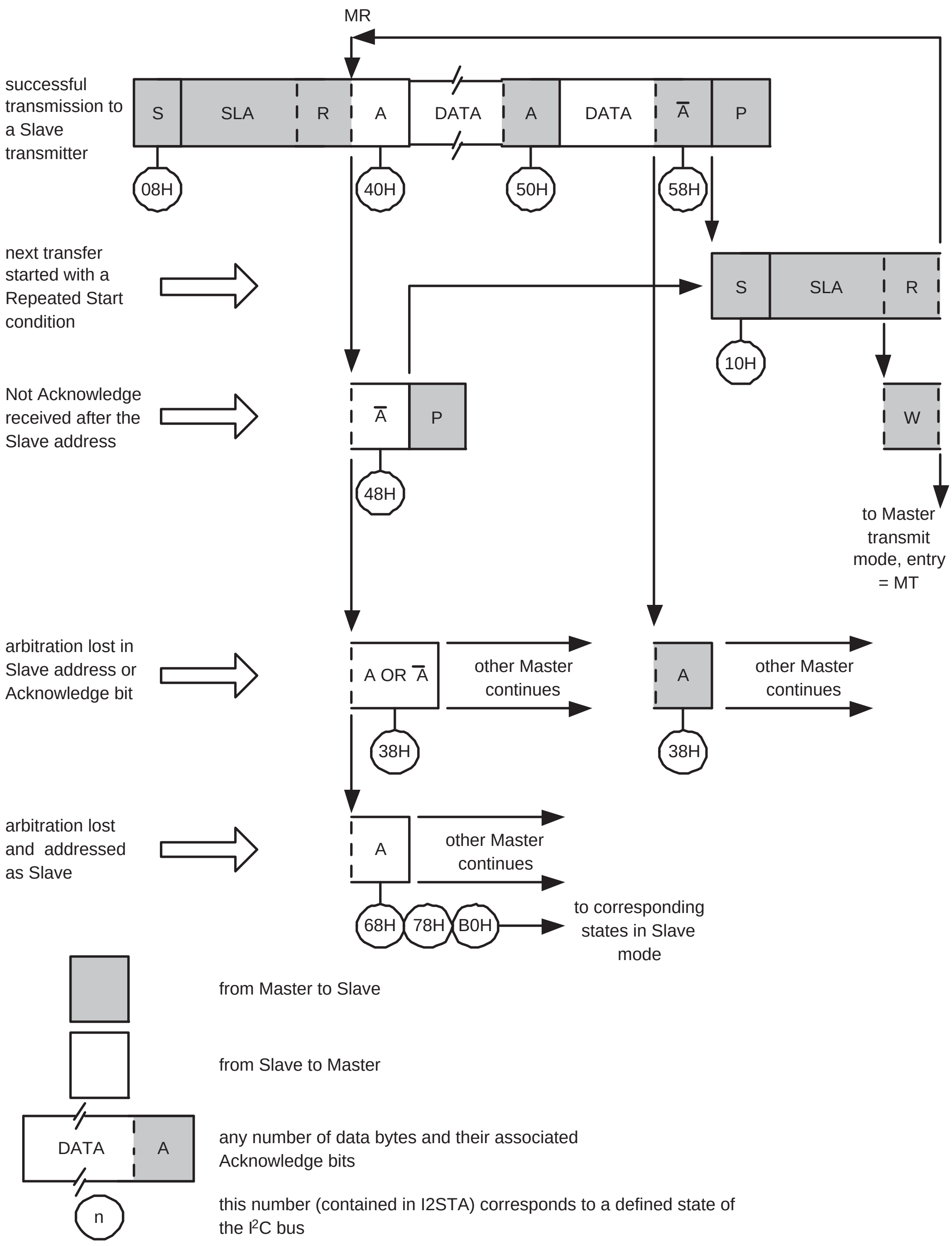

MR
successful transmission to a Slave transmitter
S
SLA
R
A
DATA
A
DATA
Ā
P
08H
40H
50H
58H
next transfer started with a Repeated Start condition
S
SLA
R
10H
W
to Master transmit mode, entry = MT
Not Acknowledge received after the Slave address
Ā
P
48H
arbitration lost in Slave address or Acknowledge bit
A OR Ā
other Master continues
38H
A
other Master continues
38H
arbitration lost and addressed as Slave
A
other Master continues
68H
78H
B0H
to corresponding states in Slave mode
from Master to Slave
from Slave to Master
DATA
A
any number of data bytes and their associated Acknowledge bits
n
this number (contained in I2STA) corresponds to a defined state of the I²C bus

Fig 35. Format and states in the Master Receiver mode

13.11.3 Slave Receiver mode

In the slave receiver mode, a number of data bytes are received from a master transmitter (see Figure 36). To initiate the slave receiver mode, I2ADR and I2CON must be loaded as follows:

Table 235. I2C0ADR and I2C1ADR usage in Slave Receiver mode

Bit	7	6	5	4	3	2	1	0
Symbol	own slave 7-bit address							GC

The upper 7 bits are the address to which the I^{2}C block will respond when addressed by a master. If the LSB (GC) is set, the I^{2}C block will respond to the General Call address (0x00); otherwise it ignores the General Call address.

Table 236. I2C0CONSET and I2C1CONSET 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^{2}C-bus rate settings do not affect the I^{2}C block in the slave mode. I2EN must be set to logic 1 to enable the I^{2}C block. The AA bit must be set to enable the I^{2}C block to acknowledge its own slave address or the General Call address. STA, STO, and SI must be reset.

When I2ADR and I2CON have been initialized, the I^{2}C 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^{2}C 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 I2STAT. 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 237. The slave receiver mode may also be entered if arbitration is lost while the I^{2}C block is in the master mode (see status 0x68 and 0x78).

If the AA bit is reset during a transfer, the I^{2}C block will return a not acknowledge (logic 1) to SDA after the next received data byte. While AA is reset, the I^{2}C block does not respond to its own slave address or a General Call address. However, the I^{2}C-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^{2}C block from the I^{2}C-bus.

Table 237. Slave Receiver mode

Status Code (I2CSTAT)	Status of the I^{2}C-bus and hardware	Application software response					Next action taken by I^{2}C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x60	Own SLA+W has been received; ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT 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 I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT 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 I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT 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 I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x80	Previously addressed with own SLV 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 I2ADR[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 I2ADR[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 237. Slave Receiver mode *…continued*

Status Code (I2CSTAT)	Status of the I²C-bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			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 I2ADR[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 I2ADR[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 SLV/REC or SLV/TRX.	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 I2ADR[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 I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.

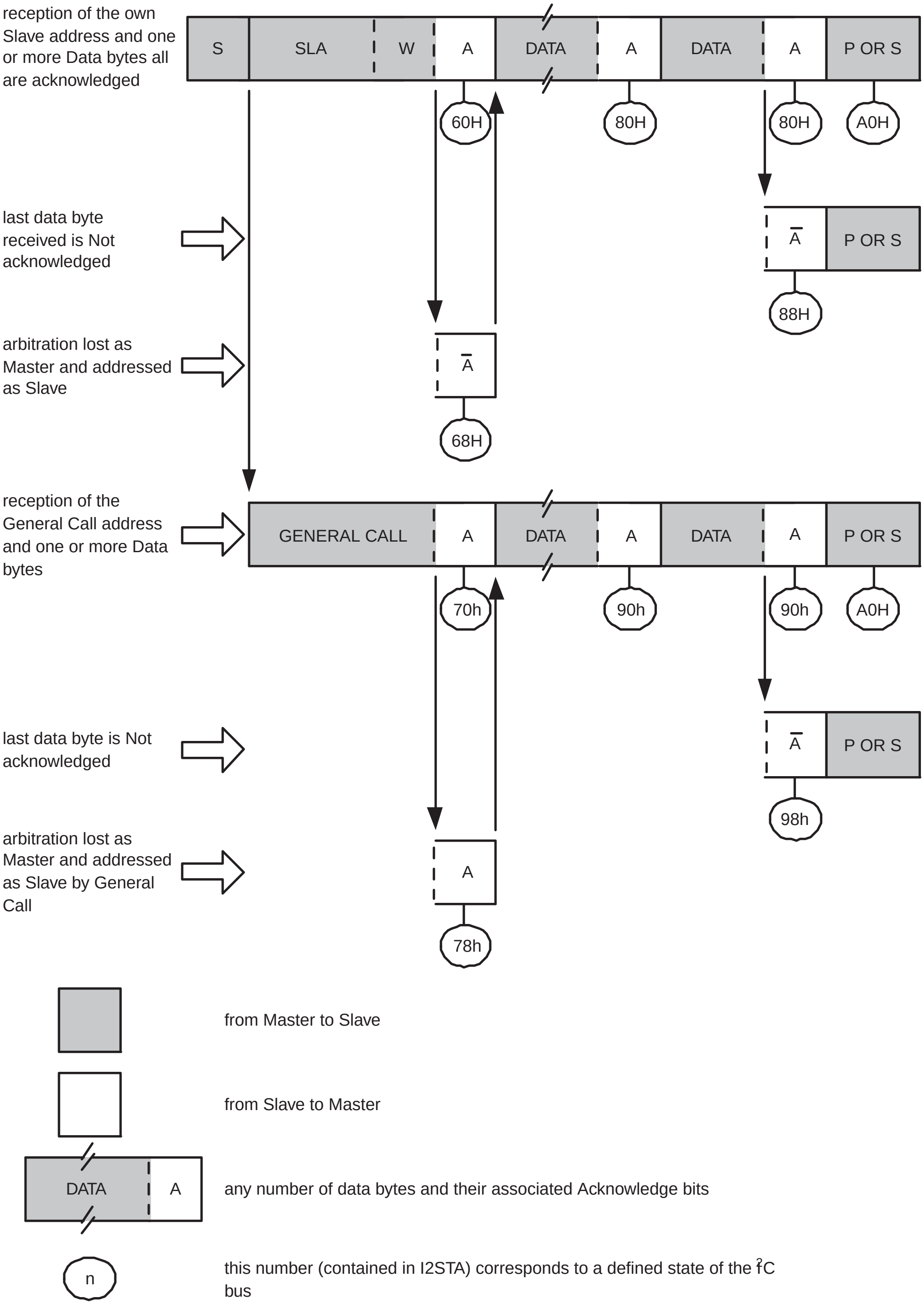

reception of the own Slave address and one or more Data bytes all are acknowledged
S
SLA
W
A
DATA
A
DATA
A
P OR S
60H
80H
80H
A0H
last data byte received is Not acknowledged
Ā
P OR S
88H
arbitration lost as Master and addressed as Slave
Ā
68H
reception of the General Call address and one or more Data bytes
GENERAL CALL
A
DATA
A
DATA
A
P OR S
70h
90h
90h
A0H
last data byte is Not acknowledged
Ā
P OR S
98h
arbitration lost as Master and addressed as Slave by General Call
A
78h
from Master to Slave
from Slave to Master
DATA
A
any number of data bytes and their associated Acknowledge bits
n
this number (contained in I2STA) corresponds to a defined state of the I²C bus

Fig 36. Format and states in the Slave Receiver mode

13.11.4 Slave Transmitter mode

In the slave transmitter mode, a number of data bytes are transmitted to a master receiver (see Figure 37). Data transfer is initialized as in the slave receiver mode. When I2ADR and I2CON 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 I2STAT. 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 238. 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.

Table 238. Slave Transmitter mode

Status Code (I2CSTAT)	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From I2DAT	To I2CON				
			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 I2DAT 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 I2DAT has been transmitted; NOT ACK has been received.	No I2DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		No I2DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1.
		No I2DAT 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 I2DAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0xC8	Last data byte in I2DAT has been transmitted (AA = 0); ACK has been received.	No I2DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		No I2DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1.
		No I2DAT 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 I2DAT action	1	0	0	01	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.

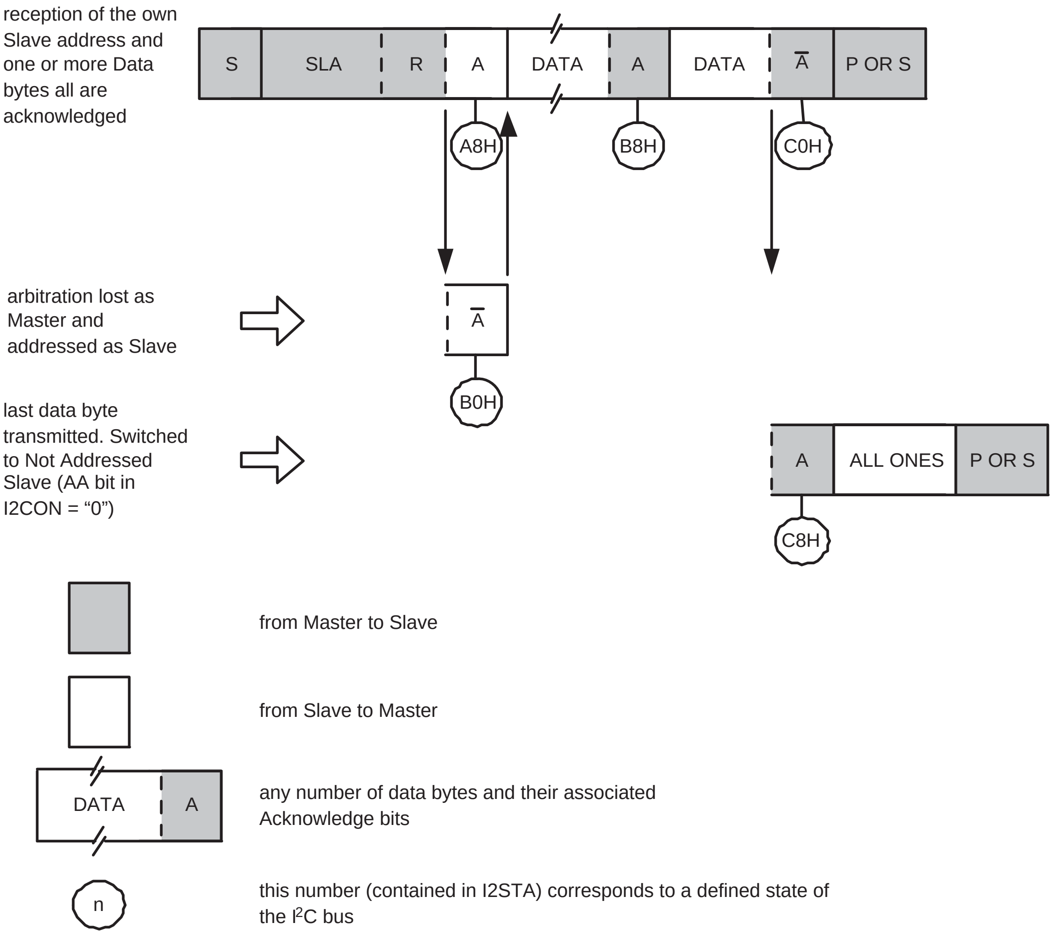

reception of the own Slave address and one or more Data bytes all are acknowledged
S
SLA
R
A
DATA
A
DATA
A̅
P OR S
A8H
B8H
C0H
arbitration lost as Master and addressed as Slave
A̅
B0H
last data byte transmitted. Switched to Not Addressed Slave (AA bit in I2CON = “0”)
A
ALL ONES
P OR S
C8H
from Master to Slave
from Slave to Master
DATA
A
any number of data bytes and their associated Acknowledge bits
n
this number (contained in I2STA) corresponds to a defined state of the I²C bus

Fig 37. Format and states in the Slave Transmitter mode

13.11.5 Miscellaneous states

There are two I2STAT codes that do not correspond to a defined I^{2}C hardware state (see Table 239). These are discussed below.

13.11.5.1 I2STAT = 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^{2}C block is not involved in a serial transfer.

13.11.5.2 I2STAT = 0x00

This status code indicates that a bus error has occurred during an I^{2}C 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^{2}C 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^{2}C block to enter the “not addressed” slave mode (a defined state) and to clear the STO flag (no other bits in I2CON are affected). The SDA and SCL lines are released (a STOP condition is not transmitted).

Table 239. Miscellaneous States

Status Code (I2CSTAT)	Status of the I^{2}C-bus and hardware	Application software response					Next action taken by I^{2}C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0xF8	No relevant state information available; SI = 0.	No I2DAT action	No I2CON 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^{2}C block to enter an undefined state.	No I2DAT 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^{2}C block is switched to the not addressed SLV mode. STO is reset.

13.11.6 Some special cases

The I^{2}C 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^{2}C-bus
- I^{2}C-bus obstructed by a LOW level on SCL or SDA
- Bus error

13.11.6.1 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 38). Until this occurs, arbitration is not lost by either master since they were both transmitting the same data.

If the I^{2}C hardware detects a Repeated START condition on the I^{2}C-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^{2}C block will transmit a normal START condition (state 0x08), and a retry of the total serial data transfer can commence.

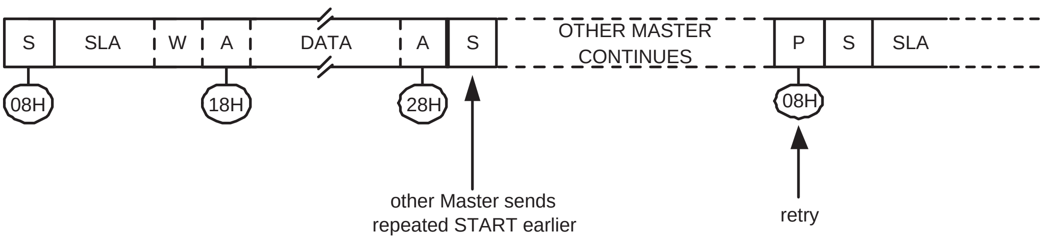


Fig 38. Simultaneous Repeated START conditions from two masters

13.11.6.2 Data transfer after loss of arbitration

Arbitration may be lost in the master transmitter and master receiver modes (see Figure 32). Loss of arbitration is indicated by the following states in I2STAT; 0x38, 0x68, 0x78, and 0xB0 (see Figure 34 and Figure 35).

If the STA flag in I2CON 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.

13.11.6.3 Forced access to the I^{2}C-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^{2}C-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^{2}C-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^{2}C hardware behaves as if a STOP condition was received and is able to transmit a START condition. The STO flag is cleared by hardware (see Figure 39).

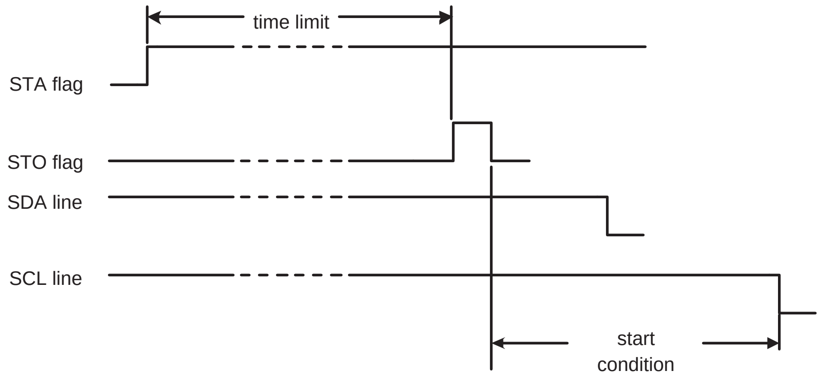
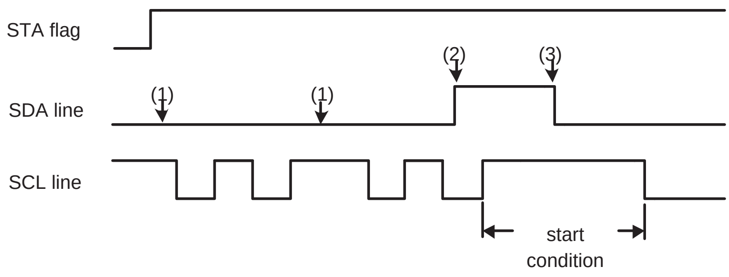


Fig 39. Forced access to a busy I^{2}C-bus

13.11.6.4 I^{2}C-bus obstructed by a LOW level on SCL or SDA

An I^{2}C-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 (see Figure 40). The I^{2}C 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 I^{2}C 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 40. Recovering from a bus obstruction caused by a LOW level on SDA

13.11.6.5 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 I^{2}C 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 I^{2}C 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 239.

13.11.7 I^{2}C state service routines

This section provides examples of operations that must be performed by various I^{2}C state service routines. This includes:

- Initialization of the I^{2}C block after a Reset.
- I^{2}C Interrupt Service
- The 26 state service routines providing support for all four I^{2}C operating modes.

13.11.8 Initialization

In the initialization example, the I^{2}C 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:

- I2ADR is loaded with the part's own slave address and the General Call bit (GC)
- The I^{2}C interrupt enable and interrupt priority bits are set
- The slave mode is enabled by simultaneously setting the I2EN and AA bits in I2CON and the serial clock frequency (for master modes) is defined by is defined by loading the I2SCLH and I2SCLL registers. The master routines must be started in the main program.

The I^{2}C hardware now begins checking the I^{2}C-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 I2STAT is loaded with the appropriate state information.

13.11.9 I^{2}C interrupt service

When the I^{2}C interrupt is entered, I2STAT contains a status code which identifies one of the 26 state services to be executed.

13.11.10 The state service routines

Each state routine is part of the I^{2}C interrupt routine and handles one of the 26 states.

13.11.11 Adapting state services to an application

The state service examples show the typical actions that must be performed in response to the 26 I^{2}C state codes. If one or more of the four I^{2}C 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 timeout during I^{2}C operations, in order to trap an inoperative bus or a lost service routine.

13.12 Software example

13.12.1 Initialization routine

Example to initialize I^{2}C Interface as a Slave and/or Master.

1. Load I2ADR with own Slave Address, enable General Call recognition if needed.
2. Enable I^{2}C interrupt.
3. Write 0x44 to I2CONSET to set the I2EN and AA bits, enabling Slave functions. For Master only functions, write 0x40 to I2CONSET.

13.12.2 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 I2CONSET 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

13.12.3 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 I2CONSET 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

13.12.4 I^{2}C interrupt routine

Determine the I^{2}C state and which state routine will be used to handle it.

1. Read the I^{2}C status from I2STA.
2. Use the status value to branch to one of 26 possible state routines.

13.12.5 Non mode specific states

13.12.5.1 State: 0x00

Bus Error. Enter not addressed Slave mode and release bus.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

13.12.5.2 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.

13.12.5.3 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 I2DAT.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR 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

13.12.5.4 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 I2DAT.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR 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

13.12.6 Master Transmitter states

13.12.6.1 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 I2DAT with first data byte from Master Transmit buffer.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Increment Master Transmit buffer pointer.
5. Exit

13.12.6.2 State: 0x20

Slave Address + Write has been transmitted, NOT ACK has been received. A STOP condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

13.12.6.3 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 I2CONSET to set the STO and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Exit
5. Load I2DAT with next data byte from Master Transmit buffer.

6. Write 0x04 to I2CONSET to set the AA bit.
7. Write 0x08 to I2CONCLR to clear the SI flag.
8. Increment Master Transmit buffer pointer
9. Exit

13.12.6.4 State: 0x30

Data has been transmitted, NOT ACK received. A STOP condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

13.12.6.5 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 I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

13.12.7 Master Receive states

13.12.7.1 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 I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

13.12.7.2 State: 0x48

Slave Address + Read has been transmitted, NOT ACK has been received. A STOP condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

13.12.7.3 State: 0x50

Data has been received, ACK has been returned. Data will be read from I2DAT. 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 I2DAT into Master Receive buffer.
2. Decrement the Master data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.

4. Exit
5. Write 0x04 to I2CONSET to set the AA bit.
6. Write 0x08 to I2CONCLR to clear the SI flag.
7. Increment Master Receive buffer pointer
8. Exit

13.12.7.4 State: 0x58

Data has been received, NOT ACK has been returned. Data will be read from I2DAT. A STOP condition will be transmitted.

1. Read data byte from I2DAT into Master Receive buffer.
2. Write 0x14 to I2CONSET to set the STO and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Exit

13.12.8 Slave Receiver states

13.12.8.1 State: 0x60

Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

13.12.8.2 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 I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit.

13.12.8.3 State: 0x70

General call has been received, ACK has been returned. Data will be received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.

4. Initialize Slave data counter.
5. Exit

13.12.8.4 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 I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

13.12.8.5 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 I2DAT 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 I2CONCLR to clear the SI flag and the AA bit.
4. Exit.
5. Write 0x04 to I2CONSET to set the AA bit.
6. Write 0x08 to I2CONCLR to clear the SI flag.
7. Increment Slave Receive buffer pointer.
8. Exit

13.12.8.6 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 I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

13.12.8.7 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 I2DAT into the Slave Receive buffer.
2. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
3. Exit

13.12.8.8 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 I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

13.12.8.9 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 I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

13.12.9 Slave Transmitter states

13.12.9.1 State: 0xA8

Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received.

1. Load I2DAT from Slave Transmit buffer with first data byte.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

13.12.9.2 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 I2DAT from Slave Transmit buffer with first data byte.
2. Write 0x24 to I2CONSET to set the STA and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

13.12.9.3 State: 0xB8

Data has been transmitted, ACK has been received. Data will be transmitted, ACK bit will be received.

1. Load I2DAT from Slave Transmit buffer with data byte.

2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Increment Slave Transmit buffer pointer.
5. Exit

13.12.9.4 State: 0xC0

Data has been transmitted, NOT ACK has been received. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit.

13.12.9.5 State: 0xC8

The last data byte has been transmitted, ACK has been received. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

UM10375

Chapter 14: LPC13xx SSP0/1

Rev. 5 — 21 June 2012 User manual

14.1 How to read this chapter

The SSP0 block is identical for all LPC13xx parts. The SSP1 block is available on part LPC1313FBD48/01 only.

14.2 Basic configuration

The SSP is configured using the following registers:

1. Pins: The SSP pins must be configured in the IOCONFIG register block (Section 7.4). The SCK0 function must also be configured in the IOCON_SCK0_LOC register (Section 7.4.43).
2. Power: In the SYSAHBCLKCTRL register, set bit 11 (Table 25).
3. Peripheral clock: Enable the SSP peripheral clock by writing to the SSP0CLKDIV register (see Table 26 and Table 28).
4. Reset: Before accessing the SSP block, ensure that the SSP_RST_N bits (bit 0 and bit 2) in the PRESETCTRL register (Table 9) are set to 1. This de-asserts the reset signal to the SSP block.

14.3 Features

- Compatible with Motorola SPI, 4-wire TI SSI, and National Semiconductor Microwire buses.
- Synchronous Serial Communication.
- Supports master or slave operation.
- Eight frame FIFOs for both transmit and receive.
- 4-bit to 16-bit frame.

14.4 General 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 bits to 16 bits 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.

The LPC13xx has one Synchronous Serial Port controller.

14.5 Pin description

Table 240. SSP pin descriptions

Pin name	Type	Interface pin name/function			Pin description
		SPI	SSI	Microwire	
SCK0/1	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).
SSEL0/1	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.

MISO0/1	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 SSP0 is a slave, serial data is output on this signal. When the SSP0 is a master, it clocks in serial data from this signal. When the SSP0 is a slave and is not selected by FS/SSEL, it does not drive this signal (leaves it in high-impedance state).

MOSI0/1	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 SSP0 is a master, it outputs serial data on this signal. When the SSP0 is a slave, it clocks in serial data from this signal.

14.6 Clocking and power control

The clocks and power to the SSP0 and SSP1 blocks are controlled by the following registers:

1. The SSP0/1 block can be enabled or disabled through the SYSAHBCLKCTRL register (see Table 25).
2. The SSP0/1_PCLK are enabled in the SSP0 and SSP1 clock divider registers (see Table 26 and Table 28). This clock is used by the SSP clock prescaler (Table 247).

Remark: Before accessing the SSP block, ensure that the SSP0/1_RST_N bits (bit 0 and bit 2) in the PRESETCTRL register (Table 9) are set to 1. This de-asserts the reset signal to the SSP block.

14.7 Register description

The register addresses of the SSP0/1 controllers are shown in Table 241.

Table 241. Register overview: SSP0 (base address 0x4004 0000)

Name	Access	Address offset	Description	Reset value[1]
SSP0CR0	R/W	0x000	Control Register 0. Selects the serial clock rate, bus type, and data size.	0
SSP0CR1	R/W	0x004	Control Register 1. Selects master/slave and other modes.	0
SSP0DR	R/W	0x008	Data Register. Writes fill the transmit FIFO, and reads empty the receive FIFO.	0
SSP0SR	RO	0x00C	Status Register.	0x0000 0003
SSP0CPSR	R/W	0x010	Clock Prescale Register.	0
SSP0IMSC	R/W	0x014	Interrupt Mask Set and Clear Register.	0
SSP0RIS	RO	0x018	Raw Interrupt Status Register.	0x0000 0008
SSP0MIS	RO	0x01C	Masked Interrupt Status Register.	0
SSP0ICR	WO	0x020	SSPICR Interrupt Clear Register.	NA

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

Table 242. Register overview: SSP1 (base address 0x4005 8000)

Name	Access	Address offset	Description	Reset value[1]
SSP1CR0	R/W	0x000	Control Register 0. Selects the serial clock rate, bus type, and data size.	0
SSP1CR1	R/W	0x004	Control Register 1. Selects master/slave and other modes.	0
SSP1DR	R/W	0x008	Data Register. Writes fill the transmit FIFO, and reads empty the receive FIFO.	0
SSP1SR	RO	0x00C	Status Register.	0x0000 0003

Table 242. Register overview: SSP1 (base address 0x4005 8000)

Name	Access	Address offset	Description	Reset value[1]
SSP1CPSR	R/W	0x010	Clock Prescale Register.	0
SSP1IMSC	R/W	0x014	Interrupt Mask Set and Clear Register.	0
SSP1RIS	RO	0x018	Raw Interrupt Status Register.	0x0000 0008
SSP1MIS	RO	0x01C	Masked Interrupt Status Register.	0
SSP1ICR	WO	0x020	SSPICR Interrupt Clear Register.	NA

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

14.7.1 SSP Control Register 0

This register controls the basic operation of the SSP controller.

Table 243: SSP Control Register 0 (SSP0CR0 - address 0x4004 0000, SSP1CR0 - address 0x4005 8000) 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.	
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.	-

14.7.2 SSP Control Register 1

This register controls certain aspects of the operation of the SSP controller.

Table 244: SSP Control Register 1 (SSP0CR1 - address 0x4004 0004, SSP1CR1 - address 0x4005 8004) 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

14.7.3 SSP Data Register

Software can write data to be transmitted to this register, and read data that has been received.

Table 245: SSP Data Register (SSP0DR - address 0x4004 0008, SSP1DR - address 0x4005 8008) 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.	-

14.7.4 SSP Status Register

This read-only register reflects the current status of the SSP controller.

Table 246: SSP Status Register (SSP0SR - address 0x4004 000C, SSP1SR - address 0x4005 800C) 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 SSP0 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

14.7.5 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 SSP0CR0, to determine the bit clock.

Table 247: SSP Clock Prescale Register (SSP0CPSR - address 0x4004 0010, SSP1CPSR - address 0x4005 8010) 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, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Important: the SSP0CPSR 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 26 or Table 28. The content of the SSP0CPSR register is not relevant.

In master mode, $CPSDVSR_{min}$ = 2 or larger (even numbers only).

14.7.6 SSP Interrupt Mask Set/Clear Register

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 248: SSP Interrupt Mask Set/Clear register (SSP0IMSC - address 0x4004 0014, SSP1IMSC - address 0x4005 8014) 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

14.7.7 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 SSP0IMSC.

Table 249: Raw Interrupt Status register (SSP0RIS - address 0x4004 0018, SSP1RIS - address 0x4005 8018) 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

14.7.8 SSP Masked Interrupt Status Register

This read-only register contains a 1 for each interrupt condition that is asserted and enabled in the IMSC register. When an SSP interrupt occurs, the interrupt service routine should read this register to determine the causes of the interrupt.

Table 250: SSP Masked Interrupt Status register (SSP0MIS - address 0x4004 001C, SSP1MIS - address 0x4005 801C) 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

14.7.9 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 SSP0IMSC.

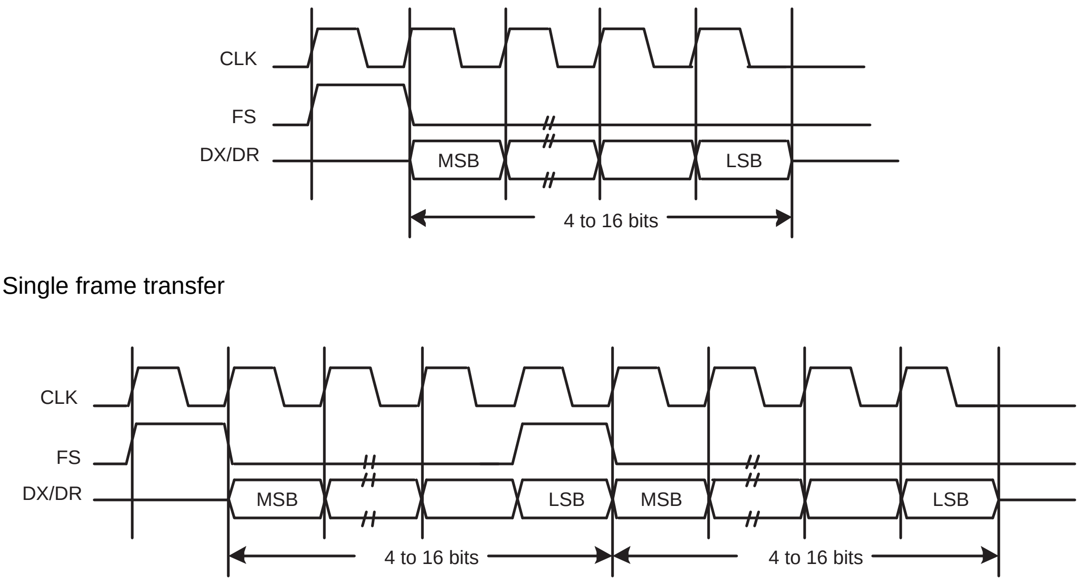
Table 251: SSP Interrupt Clear Register (SSP0ICR - address 0x4004 0020, SSP1ICR - address 0x4005 8020) bit description

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-bit 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

14.8 Functional description

14.8.1 Texas Instruments synchronous serial frame format

Figure 41 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 41. 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.

14.8.2 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.

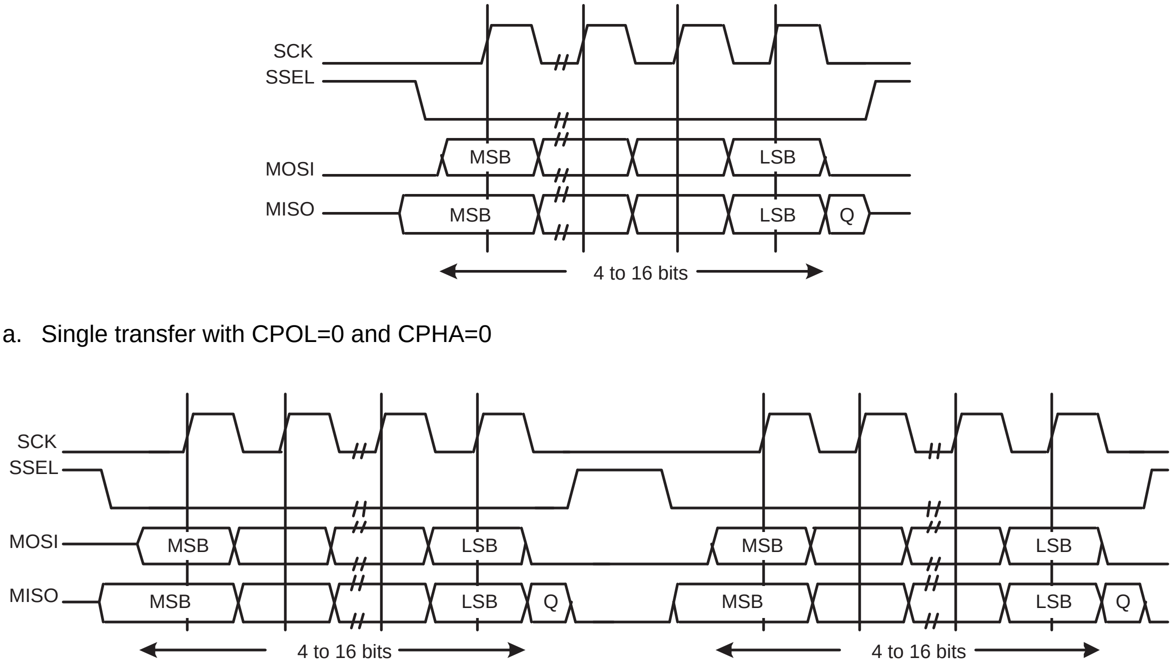
14.8.2.1 Clock Polarity (CPOL) and Phase (CPHA) control

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.

14.8.2.2 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 42.



a. Single transfer with CPOL=0 and CPHA=0

b. Continuous transfer with CPOL=0 and CPHA=0

Fig 42. 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 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.

14.8.2.3 SPI format with CPOL=0,CPHA=1

The transfer signal sequence for SPI format with CPOL = 0, CPHA = 1 is shown in Figure 43, which covers both single and continuous transfers.

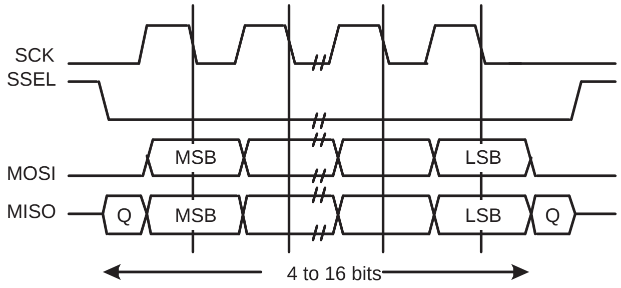


Fig 43. SPI frame format with CPOL=0 and CPHA=1

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.

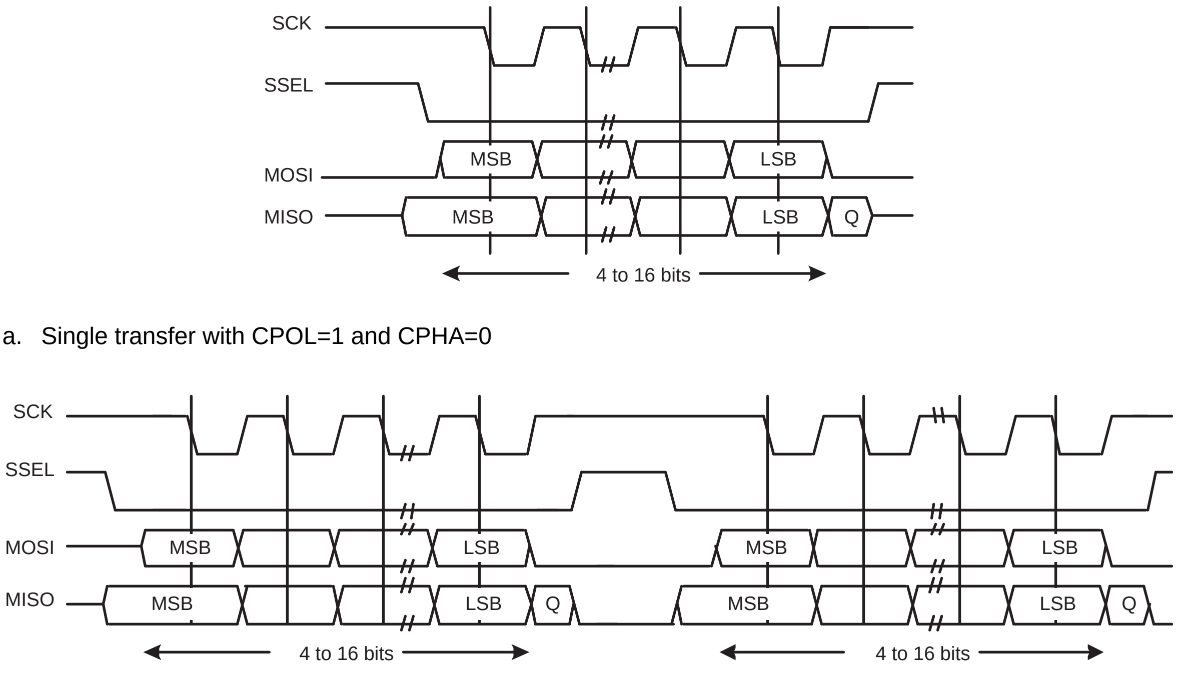
Data is then captured on the falling edges and propagated on the rising edges of the SCK signal.

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.

14.8.2.4 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 44.



a. Single transfer with CPOL=1 and CPHA=0

b. Continuous transfer with CPOL=1 and CPHA=0

Fig 44. 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.

14.8.2.5 SPI format with CPOL = 1,CPHA = 1

The transfer signal sequence for SPI format with CPOL = 1, CPHA = 1 is shown in Figure 45, which covers both single and continuous transfers.

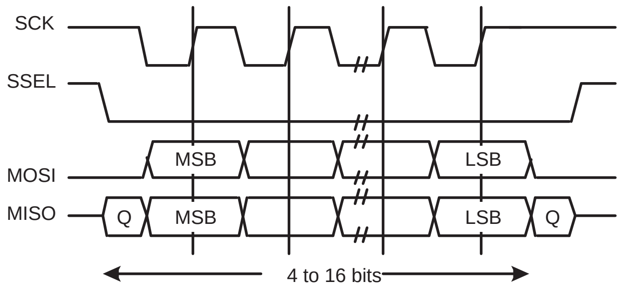


Fig 45. SPI Frame Format with CPOL = 1 and CPHA = 1

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.

14.8.3 Semiconductor Microwire frame format

Figure 46 shows the Microwire frame format for a single frame. Figure 47 shows the same format when back-to-back frames are transmitted.

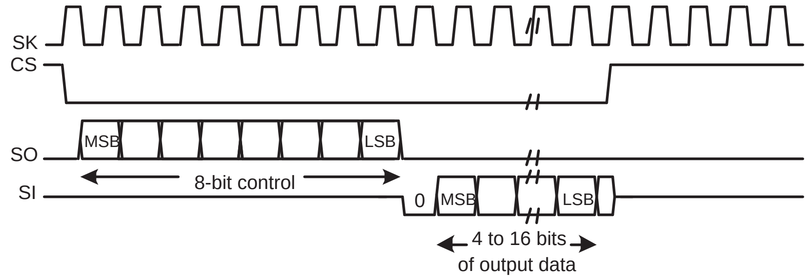


Fig 46. Microwire frame format (single transfer)

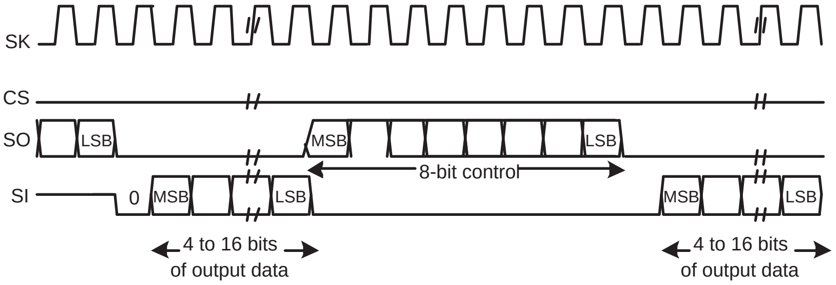


Fig 47. Microwire frame format (continuous transfers)

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 bits.

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.

14.8.3.1 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 48 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.

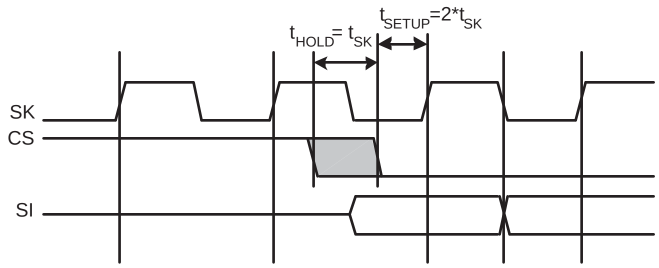


Fig 48. Microwire frame format setup and hold details

UM10375

Chapter 15: LPC13xx 16-bit timer/counters (CT16B0/1)

Rev. 5 — 21 June 2012 User manual

15.1 How to read this chapter

The 16-bit timer blocks are identical for all LPC13xx parts.

15.2 Basic configuration

The CT16B0/1 are configured using the following registers:

1. Pins: The CT16B0/1 pins must be configured in the IOCONFIG register block (Section 7.4).
2. Power and peripheral clock: In the SYSAHBCLKCTRL register, set bit 7 and bit 8 (Table 25).

15.3 Features

- Two 16-bit counter/timers with a programmable 16-bit prescaler.
- Counter or timer operation.
- One 16-bit capture channel that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- 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.
- Up to three (CT16B0) or two (CT16B1) 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.

15.4 Applications

- Interval timer for counting internal events
- Pulse Width Demodulator via capture input
- Free-running timer
- Pulse Width Modulator via match outputs

15.5 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, three match registers on CT16B0 and two match registers on CT16B1 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.

15.6 Pin description

Table 252 gives a brief summary of each of the counter/timer related pins.

Table 252. Counter/timer pin description

Pin	Type	Description
CT16B0_CAP0 CT16B1_CAP0	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 15.8.11.
CT16B0_MAT[2:0] CT16B1_MAT[1:0]	Output	External Match Outputs of CT16B0/1: When a match register of CT16B0/1 (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.

15.7 Clocking and power control

The peripheral clocks (PCLK) to the 16-bit timers are provided by the system clock (see Figure 3). These clocks can be disabled through bit 7 and 8 in the SYSAHBCLKCTRL register (Table 25) for power savings.

15.8 Register description

The 16-bit counter/timer0 contains the registers shown in Table 253 and the 16-bit counter/timer1 contains the registers shown in Table 254. More detailed descriptions follow.

Table 253. Register overview: 16-bit counter/timer 0 CT16B0 (base address 0x4000 C000)

Name	Access	Address offset	Description	Reset value[1]
TMR16B0IR	R/W	0x000	Interrupt Register (IR). The IR can be written to clear interrupts. The IR can be read to identify which of five possible interrupt sources are pending.	0
TMR16B0TCR	R/W	0x004	Timer Control Register (TCR). The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0
TMR16B0TC	R/W	0x008	Timer Counter (TC). The 16-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0
TMR16B0PR	R/W	0x00C	Prescale Register (PR). When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	0
TMR16B0PC	R/W	0x010	Prescale Counter (PC). 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
TMR16B0MCR	R/W	0x014	Match Control Register (MCR). The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	0
TMR16B0MR0	R/W	0x018	Match Register 0 (MR0). 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
TMR16B0MR1	R/W	0x01C	Match Register 1 (MR1). See MR0 description.	0
TMR16B0MR2	R/W	0x020	Match Register 2 (MR2). See MR0 description.	0
TMR16B0MR3	R/W	0x024	Match Register 3 (MR3). See MR0 description.	0
TMR16B0CCR	R/W	0x028	Capture Control Register (CCR). 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
TMR16B0CR0	RO	0x02C	Capture Register 0 (CR0). CR0 is loaded with the value of TC when there is an event on the CT16B0_CAP0 input.	0
TMR16B0EMR	R/W	0x03C	External Match Register (EMR). The EMR controls the match function and the external match pins CT16B0_MAT[2:0].	0
-	-	0x040 - 0x06C	reserved	-
TMR16B0CTCR	R/W	0x070	Count Control Register (CTCR). The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0
TMR16B0PWMC	R/W	0x074	PWM Control Register (PWMCON). The PWMCON enables PWM mode for the external match pins CT16B0_MAT[2:0].	0

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

Table 254. Register overview: 16-bit counter/timer 1 CT16B1 (base address 0x4001 0000)

Name	Access	Address offset	Description	Reset value[1]
TMR16B1IR	R/W	0x000	Interrupt Register (IR). The IR can be written to clear interrupts. The IR can be read to identify which of five possible interrupt sources are pending.	0
TMR16B1TCR	R/W	0x004	Timer Control Register (TCR). The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0
TMR16B1TC	R/W	0x008	Timer Counter (TC). The 16-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0
TMR16B1PR	R/W	0x00C	Prescale Register (PR). When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	0
TMR16B1PC	R/W	0x010	Prescale Counter (PC). 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
TMR16B1MCR	R/W	0x014	Match Control Register (MCR). The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	0
TMR16B1MR0	R/W	0x018	Match Register 0 (MR0). 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
TMR16B1MR1	R/W	0x01C	Match Register 1 (MR1). See MR0 description.	0
TMR16B1MR2	R/W	0x020	Match Register 2 (MR2). See MR0 description.	0
TMR16B1MR3	R/W	0x024	Match Register 3 (MR3). See MR0 description.	0
TMR16B1CCR	R/W	0x028	Capture Control Register (CCR). 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
TMR16B1CR0	RO	0x02C	Capture Register 0 (CR0). CR0 is loaded with the value of TC when there is an event on the CT16B1_CAP0 input.	0
TMR16B1EMR	R/W	0x03C	External Match Register (EMR). The EMR controls the match function and the external match pins CT16B1_MAT[1:0].	0
-	-	0x040 - 0x06C	reserved	-
TMR16B1CTCR	R/W	0x070	Count Control Register (CTCR). The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0
TMR16B1PWMC	R/W	0x074	PWM Control Register (PWMCON). The PWMCON enables PWM mode for the external match pins CT16B1_MAT[1:0].	0

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

15.8.1 Interrupt Register (TMR16B0IR and TMR16B1IR)

The Interrupt Register (IR) consists of four bits for the match interrupts and one bit for the capture interrupt. If an interrupt is generated then the corresponding bit in the IR will be HIGH. Otherwise, the bit will be LOW. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

Table 255. Interrupt Register (TMR16B0IR - address 0x4000 C000 and TMR16B1IR - address 0x4001 0000) 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
31:5	-	Reserved	-

15.8.2 Timer Control Register (TMR16B0TCR and TMR16B1TCR)

The Timer Control Register (TCR) is used to control the operation of the counter/timer.

Table 256. Timer Control Register (TMR16B0TCR - address 0x4000 C004 and TMR16B1TCR - address 0x4001 0004) bit description

Bit	Symbol	Description	Reset value
0	CEN	When one, the Timer Counter and Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	CRESET	When one, 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.	0
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

15.8.3 Timer Counter (TMR16B0TC - address 0x4000 C008 and TMR16B1TC - address 0x4001 0008)

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 through 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 257: Timer counter registers (TMR16B0TC, address 0x4000 C008 and TMR16B1TC 0x4001 0008) bit description

Bit	Symbol	Description	Reset value
15:0	TC	Timer counter value.	0
31:16	-	Reserved.	-

15.8.4 Prescale Register (TMR16B0PR - address 0x4000 C00C and TMR16B1PR - address 0x4001 000C)

The 16-bit Prescale Register specifies the maximum value for the Prescale Counter.

Table 258: Prescale registers (TMR16B0PR, address 0x4000 C00C and TMR16B1PR 0x4001 000C) bit description

Bit	Symbol	Description	Reset value
15:0	PR	Prescale max value.	0
31:16	-	Reserved.	-

15.8.5 Prescale Counter register (TMR16B0PC - address 0x4000 C010 and TMR16B1PC - address 0x4001 0010)

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 259: Prescale counter registers (TMR16B0PC, address 0x4001 C010 and TMR16B1PC 0x4000 0010) bit description

Bit	Symbol	Description	Reset value
15:0	PC	Prescale counter value.	0
31:16	-	Reserved.	-

15.8.6 Match Control Register (TMR16B0MCR and TMR16B1MCR)

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 260.

Table 260. Match Control Register (TMR16B0MCR - address 0x4000 C014 and TMR16B1MCR - address 0x4001 0014) 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	

Table 260. Match Control Register (TMR16B0MCR - address 0x4000 C014 and TMR16B1MCR - address 0x4001 0014) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
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

15.8.7 Match Registers (TMR16B0MR0/1/2/3 - addresses 0x4000 C018/1C/20/24 and TMR16B1MR0/1/2/3 - addresses 0x4001 0018/1C/20/24)

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 261: Match registers (TMR16B0MR0 to 3, addresses 0x4000 C018 to 24 and TMR16B1MR0 to 3, addresses 0x4001 0018 to 24) bit description

Bit	Symbol	Description	Reset value
15:0	MATCH	Timer counter match value.	0
31:16	-	Reserved.	-

15.8.8 Capture Control Register (TMR16B0CCR and TMR16B1CCR)

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.

Table 262. Capture Control Register (TMR16B0CCR - address 0x4000 C028 and TMR16B1CCR - address 0x4001 0028) 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	
31:3	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

15.8.9 Capture Register (CT16B0CR0 - address 0x4000 C02C and CT16B1CR0 - address 0x4001 002C)

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 263: Capture registers (TMR16B0CR0, address 0x4000 C02C and TMR16B1CR0, address 0x4001 002C) bit description

Bit	Symbol	Description	Reset value
15:0	CAP	Timer counter capture value.	0
31:16	-	Reserved.	-

15.8.10 External Match Register (TMR16B0EMR and TMR16B1EMR)

The External Match Register provides both control and status of the external match channels and external match pins CT16B0_MAT[2:0] and CT16B1_MAT[1:0].

If the match outputs are configured as PWM output in the PWMCON registers (Section 15.8.12), the function of the external match registers is determined by the PWM rules (Section 15.8.13 "Rules for single edge controlled PWM outputs" on page 279).

Table 264. External Match Register (TMR16B0EMR - address 0x4000 C03C and TMR16B1EMR - address 0x4001 003C) 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_MAT1/CT16B1_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 match channel 2, 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. Note that on counter/timer 0 this match channel is not pinned out. This bit is driven to the CT16B1_MAT2 pin 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 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. There is no output pin available for this channel on either of the 16-bit timers.	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 (CT16Bn_MATm pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT16Bn_MATm 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 (CT16Bn_MATm pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT16Bn_MATm pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	

Table 264. External Match Register (TMR16B0EMR - address 0x4000 C03C and TMR16B1EMR - address 0x4001 003C) bit description

Bit	Symbol	Value	Description	Reset value
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 (CT16Bn_MATm pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT16Bn_MATm 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 (CT16Bn_MATm pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT16Bn_MATm 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 265. 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.

15.8.11 Count Control Register (TMR16B0CTCR and TMR16B1CTCR)

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/LOW levels on the same CAP input in this case can not be shorter than 1/(2 × PCLK).

Table 266. Count Control Register (TMR16B0CTCR - address 0x4000 C070 and TMR16B1CTCR - address 0x4001 0070) 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).	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 is sampled for clocking. **Note:** If Counter mode is selected in the CTCR register, bits 2:0 in the Capture Control Register (CCR) must be programmed as 000.	00
		0x0	CT16Bn_CAP0	
		0x1	Reserved.	
		0x2	Reserved.	
		0x0	Reserved.	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

15.8.12 PWM Control register (TMR16B0PWMC and TMR16B1PWMC)

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 timer 0, three single-edge controlled PWM outputs can be selected on the CT16B0_MAT[2:0] outputs. For timer 1, two single-edged PWM outputs can be selected on the CT16B1_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 267. PWM Control Register (TMR16B0PWMC - address 0x4000 C074 and TMR16B1PWMC- address 0x4001 0074) bit description

Bit	Symbol	Value	Description	Reset value
0	PWMEN0		PWM channel0 enable	0
		0	CT16Bn_MAT0 is controlled by EM0.	
		1	PWM mode is enabled for CT16Bn_MAT0.	

Table 267. PWM Control Register (TMR16B0PWMC - address 0x4000 C074 and TMR16B1PWMC- address 0x4001 0074) bit description

Bit	Symbol	Value	Description	Reset value
1	PWMEN1		PWM channel1 enable	0
		0	CT16Bn_MAT1 is controlled by EM1.	
		1	PWM mode is enabled for CT16Bn_MAT1.	
2	PWMEN2		PWM channel2 enable	0
		0	Match channel 2 or pin CT16B0_MAT2 is controlled by EM2. Match channel 2 is not pinned out on timer 1.	
		1	PWM mode is enabled for match channel 2 or pin CT16B0_MAT2.	
3	PWMEN3		PWM channel3 enable **Note:** It is recommended to use to set the PWM cycle because it is not pinned out.	0
		0	Match channel 3 is controlled by EM3.	
		1	PWM mode is enabled for match channel 3.	
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

15.8.13 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 serve as PWM outputs, the timer reset (MRnR) and timer stop (MRnS) bits in the Match Control Register MCR must be set to 0 except for the match register setting the PWM cycle length. For this register, set the MRnR bit to 1 to enable the timer reset when the timer value matches the value of the corresponding match register.

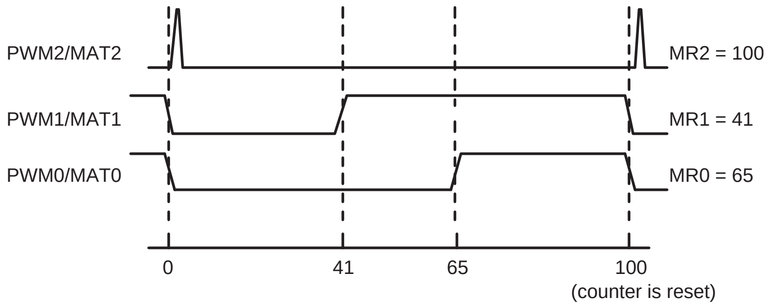


Fig 49. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

15.9 Example timer operation

Figure 50 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 51 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.

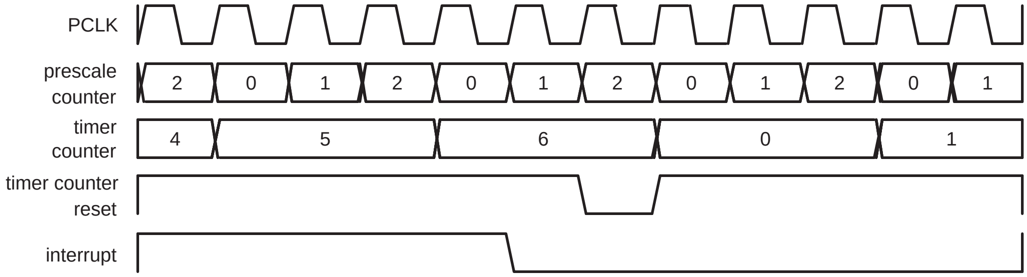


Fig 50. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

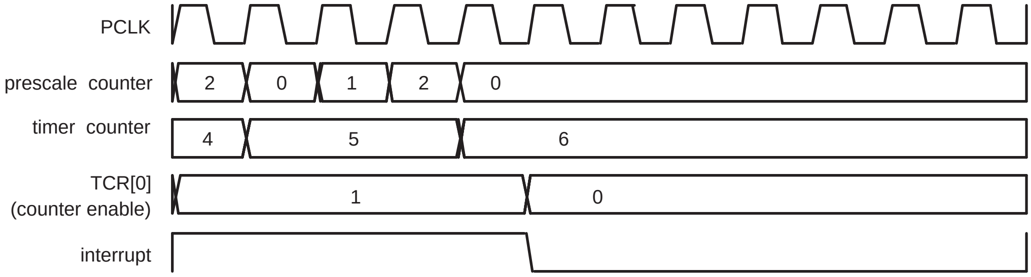


Fig 51. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

15.10 Architecture

The block diagram for counter/timer0 and counter/timer1 is shown in Figure 52.

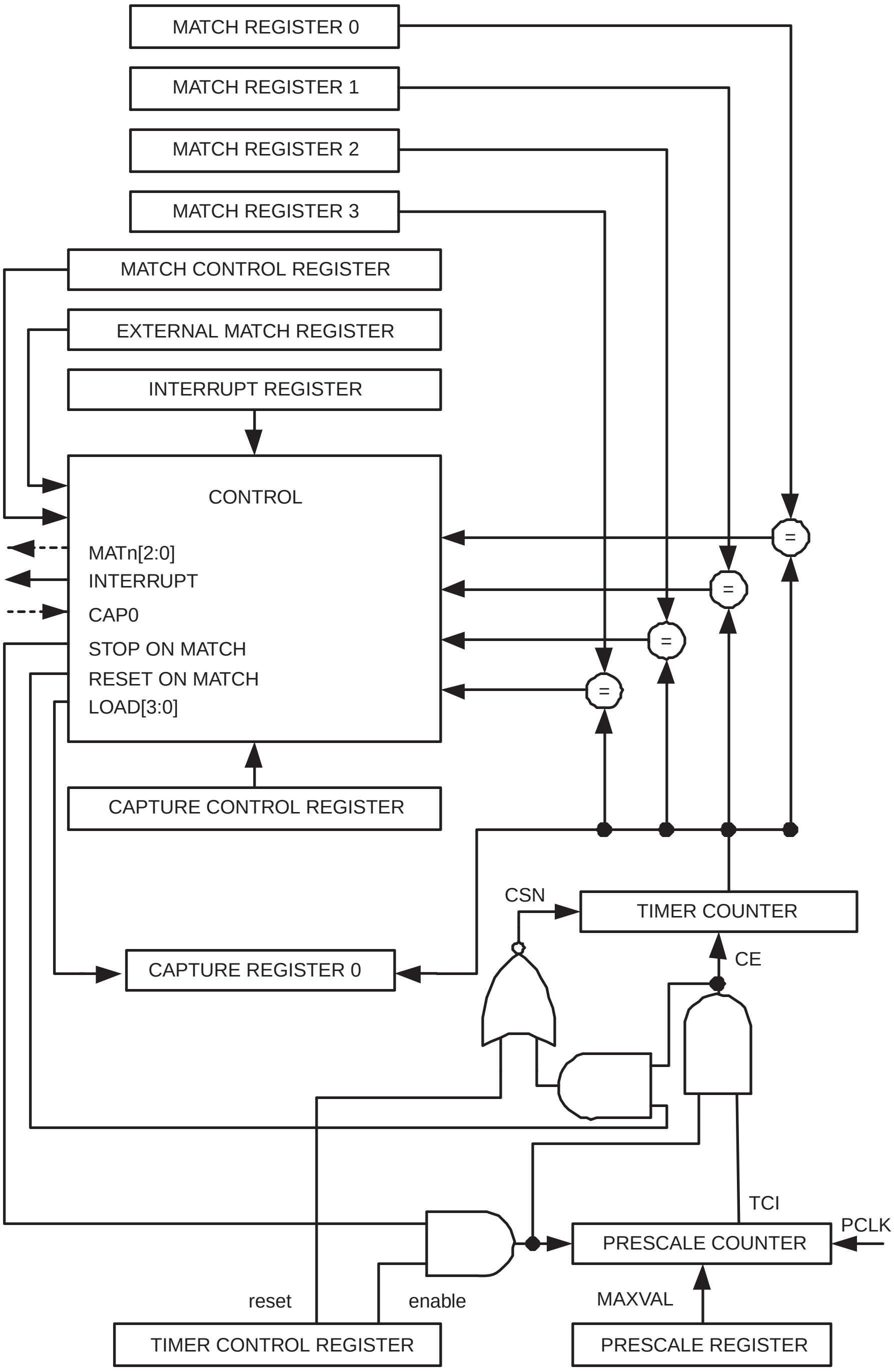


Fig 52. 16-bit counter/timer block diagram

UM10375

Chapter 16: LPC13xx 32-bit timer/counters (CT32B0/1)

Rev. 5 — 21 June 2012 User manual

16.1 How to read this chapter

The 32-bit timer blocks are identical for all LPC13xx parts.

16.2 Basic configuration

The CT32B0/1 are configured using the following registers:

1. Pins: The CT32B0/1 pins must be configured in the IOCONFIG register block (Section 7.4).
2. Power and peripheral clock: In the SYSAHBCLKCTRL register, set bit 9 and bit 10 (Table 25).

16.3 Features

- Two 32-bit counter/timers with a programmable 32-bit prescaler.
- Counter or Timer operation.
- One 32-bit capture channel that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- 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.

16.4 Applications

- Interval timer for counting internal events
- Pulse Width Demodulator via capture input
- Free running timer
- Pulse Width Modulator via match outputs

16.5 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, 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 the peripheral base address.

16.6 Pin description

Table 268 gives a brief summary of each of the counter/timer related pins.

Table 268. Counter/timer pin description

Pin	Type	Description
CT32B0_CAP0 CT32B1_CAP0	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 16.8.11 “Count Control Register (TMR32B0CTCR and TMR32B1TCR)” on page 291.
CT32B0_MAT[3:0] CT32B1_MAT[3:0]	Output	External Match Output of CT32B0/1: When a match register TMR32B0/1MR3:0 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.

16.7 Clocking and power control

The peripheral clocks (PCLK) to the 32-bit timers are provided by the system clock (see Figure 3). These clocks can be disabled through bits 9 and 10 in the SYSAHBCLKCTRL register (Table 25) for power savings.

16.8 Register description

32-bit counter/timer0 contains the registers shown in Table 269 and 32-bit counter/timer1 contains the registers shown in Table 270. More detailed descriptions follow.

Table 269. Register overview: 32-bit counter/timer 0 CT32B0 (base address 0x4001 4000)

Name	Access	Address offset	Description	Reset value[1]
TMR32B0IR	R/W	0x000	Interrupt Register (IR). The IR can be written to clear interrupts. The IR can be read to identify which of five possible interrupt sources are pending.	0
TMR32B0TCR	R/W	0x004	Timer Control Register (TCR). The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0
TMR32B0TC	R/W	0x008	Timer Counter (TC). The 32-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0
TMR32B0PR	R/W	0x00C	Prescale Register (PR). When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	0
TMR32B0PC	R/W	0x010	Prescale Counter (PC). 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
TMR32B0MCR	R/W	0x014	Match Control Register (MCR). The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	0
TMR32B0MR0	R/W	0x018	Match Register 0 (MR0). 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
TMR32B0MR1	R/W	0x01C	Match Register 1 (MR1). See MR0 description.	0
TMR32B0MR2	R/W	0x020	Match Register 2 (MR2). See MR0 description.	0
TMR32B0MR3	R/W	0x024	Match Register 3 (MR3). See MR0 description.	0
TMR32B0CCR	R/W	0x028	Capture Control Register (CCR). 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
TMR32B0CR0	RO	0x02C	Capture Register 0 (CR0). CR0 is loaded with the value of TC when there is an event on the CT32B0_CAP0 input.	0
TMR32B0EMR	R/W	0x03C	External Match Register (EMR). The EMR controls the match function and the external match pins CT32B0_MAT[3:0].	0
-	-	0x040 - 0x06C	reserved	-
TMR32B0CTCR	R/W	0x070	Count Control Register (CTCR). The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0
TMR32B0PWMC	R/W	0x074	PWM Control Register (PWMCON). The PWMCON enables PWM mode for the external match pins CT32B0_MAT[3:0].	0

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

Table 270. Register overview: 32-bit counter/timer 1 CT32B1 (base address 0x4001 8000)

Name	Access	Address offset	Description	Reset value[1]
TMR32B1IR	R/W	0x000	Interrupt Register (IR). The IR can be written to clear interrupts. The IR can be read to identify which of five possible interrupt sources are pending.	0
TMR32B1TCR	R/W	0x004	Timer Control Register (TCR). The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0
TMR32B1TC	R/W	0x008	Timer Counter (TC). The 32-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0
TMR32B1PR	R/W	0x00C	Prescale Register (PR). When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	0
TMR32B1PC	R/W	0x010	Prescale Counter (PC). 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
TMR32B1MCR	R/W	0x014	Match Control Register (MCR). The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	0
TMR32B1MR0	R/W	0x018	Match Register 0 (MR0). 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
TMR32B1MR1	R/W	0x01C	Match Register 1 (MR1). See MR0 description.	0
TMR32B1MR2	R/W	0x020	Match Register 2 (MR2). See MR0 description.	0
TMR32B1MR3	R/W	0x024	Match Register 3 (MR3). See MR0 description.	0
TMR32B1CCR	R/W	0x028	Capture Control Register (CCR). 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
TMR32B1CR0	RO	0x02C	Capture Register 0 (CR0). CR0 is loaded with the value of TC when there is an event on the CT32B1_CAP0 input.	0
TMR32B1EMR	R/W	0x03C	External Match Register (EMR). The EMR controls the match function and the external match pins CT32B1_MAT[3:0].	0
-	-	0x040 - 0x06C	reserved	-
TMR32B1CTCR	R/W	0x070	Count Control Register (CTCR). The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0
TMR32B1PWMC	R/W	0x074	PWM Control Register (PWMCON). The PWMCON enables PWM mode for the external match pins CT32B1_MAT[3:0].	0

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

16.8.1 Interrupt Register (TMR32B0IR and TMR32B1IR)

The Interrupt Register consists of four bits for the match interrupts and one bit for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be HIGH. Otherwise, the bit will be LOW. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

Table 271: Interrupt Register (TMR32B0IR - address 0x4001 4000 and TMR32B1IR - address 0x4001 8000) 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
31:5	-	Reserved	-

16.8.2 Timer Control Register (TMR32B0TCR and TMR32B1TCR)

The Timer Control Register (TCR) is used to control the operation of the counter/timer.

Table 272: Timer Control Register (TMR32B0TCR - address 0x4001 4004 and TMR32B1TCR - address 0x4001 8004) bit description

Bit	Symbol	Description	Reset value
0	CEN	When one, the Timer Counter and Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	CRES	When one, 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.	0
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

16.8.3 Timer Counter (TMR32B0TC - address 0x4001 4008 and TMR32B1TC - address 0x4001 8008)

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 273: Timer counter registers (TMR32B0TC, address 0x4001 4008 and TMR32B1TC 0x4001 8008) bit description

Bit	Symbol	Description	Reset value
31:0	TC	Timer counter value.	0

16.8.4 Prescale Register (TMR32B0PR - address 0x4001 400C and TMR32B1PR - address 0x4001 800C)

The 32-bit Prescale Register specifies the maximum value for the Prescale Counter.

Table 274: Prescale registers (TMR32B0PR, address 0x4001 400C and TMR32B1PR 0x4001 800C) bit description

Bit	Symbol	Description	Reset value
31:0	PR	Prescale value.	0

16.8.5 Prescale Counter Register (TMR32B0PC - address 0x4001 4010 and TMR32B1PC - address 0x4001 8010)

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 275: Prescale counter registers (TMR32B0PC, address 0x4001 4010 and TMR32B1PC 0x4001 8010) bit description

Bit	Symbol	Description	Reset value
31:0	PC	Prescale counter value.	0

16.8.6 Match Control Register (TMR32B0MCR and TMR32B1MCR)

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 276.

Table 276: Match Control Register (TMR32B0MCR - address 0x4001 4014 and TMR32B1MCR - address 0x4001 8014) 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	

All information provided in this document is subject to legal disclaimers.

Table 276: Match Control Register (TMR32B0MCR - address 0x4001 4014 and TMR32B1MCR - address 0x4001 8014) 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

16.8.7 Match Registers (TMR32B0MR0/1/2/3 - addresses 0x4001 4018/1C/20/24 and TMR32B1MR0/1/2/3 addresses 0x4001 8018/1C/20/24)

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 277: Match registers (TMR32B0MR0 to 3, addresses 0x4001 4018 to 24 and TMR32B1MR0 to 3, addresses 0x4001 8018 to 24) bit description

Bit	Symbol	Description	Reset value
31:0	MATCH	Timer counter match value.	0

16.8.8 Capture Control Register (TMR32B0CCR and TMR32B1CCR)

The Capture Control Register is used to control whether the Capture Register 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 278: Capture Control Register (TMR32B0CCR - address 0x4001 4028 and TMR32B1CCR - address 0x4001 8028) bit description

Bit	Symbol	Value	Description	Reset value
0	CAP0RE		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		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	
31:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

16.8.9 Capture Register (TMR32B0CR0 - address 0x4001 402C and TMR32B1CR0 - address 0x4001 802C)

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 279: Capture registers (TMR32B0CR0, addresses 0x4001 402C and TMR32B1CR0, addresses 0x4001 802C) bit description

Bit	Symbol	Description	Reset value
31:0	CAP	Timer counter capture value.	0

16.8.10 External Match Register (TMR32B0EMR and TMR32B1EMR)

The External Match Register provides both control and status of the external match pins CAP32Bn_MAT[3:0].

If the match outputs are configured as PWM output, the function of the external match registers is determined by the PWM rules (Section 16.8.13 “Rules for single edge controlled PWM outputs” on page 293).

Table 280: External Match Register (TMR32B0EMR - address 0x4001 403C and TMR32B1EMR - address0x4001 803C) 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/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 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/CT16B1_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/CT16B1_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 CT32B0_MAT3/CT16B1_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 (CT32Bn_MATm pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT32Bn_MATm 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 (CT32Bn_MATm pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT32Bn_MATm 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 (CT32Bn_MATm pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT32Bn_MATm pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	

Table 280: External Match Register (TMR32B0EMR - address 0x4001 403C and TMR32B1EMR - address0x4001 803C) bit description

Bit	Symbol	Value	Description	Reset value
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 (CT32Bn_MATm pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT32Bn_MATm 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 281. 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.

16.8.11 Count Control Register (TMR32B0CTCR and TMR32B1TCR)

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/LOW levels on the same CAP input in this case can not be shorter than 1/(2 × PCLK).

Table 282: Count Control Register (TMR32B0CTCR - address 0x4001 4070 and TMR32B1TCR - address 0x4001 8070) 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). Timer Mode: every rising PCLK edge	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. When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking:	00
		0x0	CT32Bn_CAP0	
		0x1	Reserved	
		0x2	Reserved	
		0x3	Reserved **Note:** If Counter mode is selected in the TnCTCR, the 3 bits for that input in the Capture Control Register (TnCCR) must be programmed as 000.	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

16.8.12 PWM Control Register (TMR32B0PWMC and TMR32B1PWMC)

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 283: PWM Control Register (TMR32B0PWMC - 0x4001 4074 and TMR32B1PWMC - 0x4001 8074) bit description

Bit	Symbol	Value	Description	Reset value
0	PWMEN0		PWM channel 0 enable	0
		0	CT32Bn_MAT0 is controlled by EM0.	
		1	PWM mode is enabled for CT32Bn_MAT0.	

Table 283: PWM Control Register (TMR32B0PWMC - 0x4001 4074 and TMR32B1PWMC - 0x4001 8074) bit description

Bit	Symbol	Value	Description	Reset value
1	PWMEN1		PWM channel 1 enable	0
		0	CT32Bn_MAT1 is controlled by EM1.	
		1	PWM mode is enabled for CT32Bn_MAT1.	
2	PWMEN2		PWM channel 2 enable	0
		0	CT32Bn_MAT2 is controlled by EM2.	
		1	PWM mode is enabled for CT32Bn_MAT2.	
3	PWMEN3		PWM channel 3 enable **Note:** It is recommended to use match channel 3 to set the PWM cycle.	0
		0	CT32Bn_MAT3 is controlled by EM3.	
		1	PWM mode is enabled for CT32Bn_MAT3.	
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

16.8.13 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 function as PWM outputs, the timer reset (MRnR) and timer stop (MRnS) bits in the Match Control Register MCR must be set to 0 except for the match register setting the PWM cycle length. For this register, set the MRnR bit to 1 to enable the timer reset when the timer value matches the value of the corresponding match register.

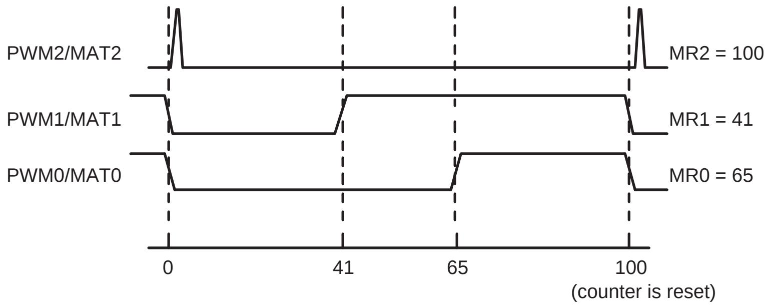


Fig 53. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

16.9 Example timer operation

Figure 54 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 55 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.

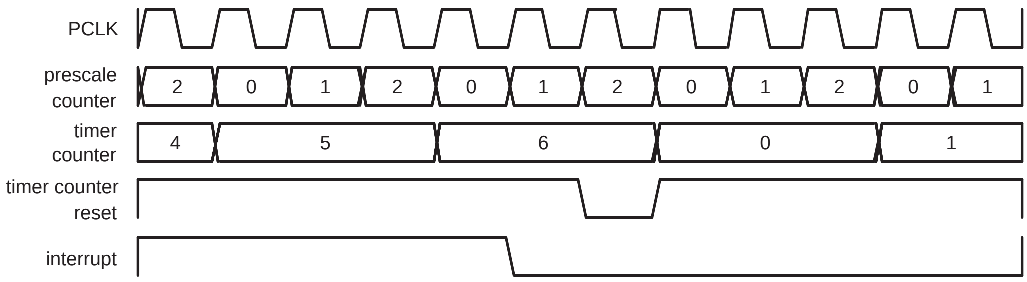


Fig 54. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

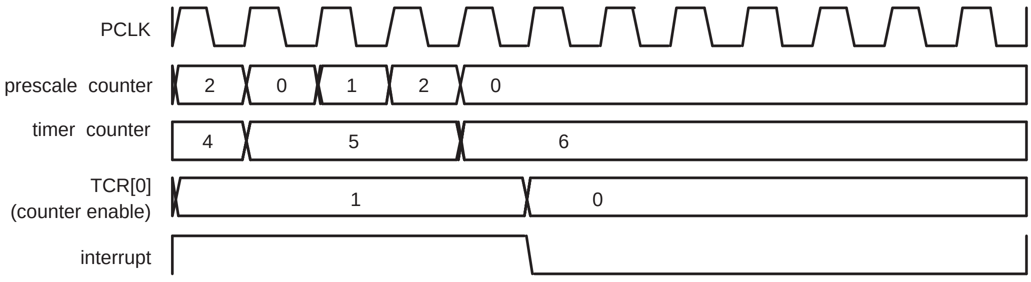


Fig 55. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

16.10 Architecture

The block diagram for 32-bit counter/timer0 and 32-bit counter/timer1 is shown in Figure 56.

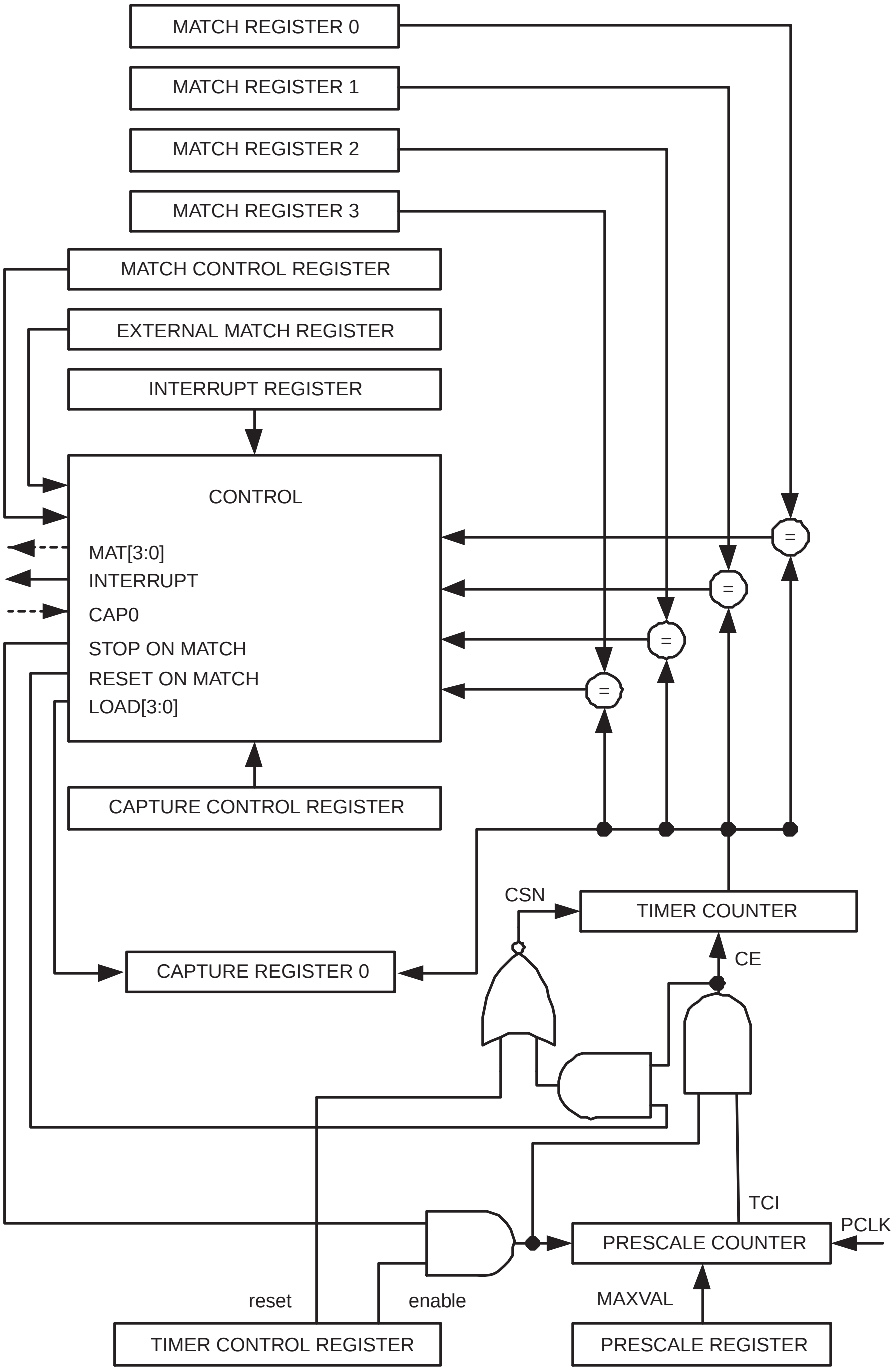


Fig 56. 32-bit counter/timer block diagram

17.1 How to read this chapter

The system tick timer (SysTick timer) is part of the ARM Cortex-M3 core and is identical for all LPC13xx parts.

17.2 Basic configuration

The system tick timer is configured using the following registers:

1. Pins: The system tick timer uses no external pins.
2. Power and peripheral clock: The system tick timer is enabled through the SysTick control register (Table 285). The system tick timer clock can be selected from the Systick timer clock divider in the system configuration block (Table 30) or the system clock (see Table 285).

17.3 Features

- 24-bit timer.
- Intended to time intervals of 10 ms.
- Uses dedicated exception vector.
- Clocked internally by the CPU system clock or dedicated system tick timer clock.

17.4 Description

The System Tick Timer is an integral part of the Cortex-M3. The System Tick Timer is intended to generate a fixed 10 millisecond interrupt for use by an operating system or other system management software.

Since the System Tick Timer is a part of the Cortex-M3, it facilitates porting of software by providing a standard timer that is available on Cortex-M3 based devices.

Refer to the *Cortex-M3 User Guide* for details.

17.5 Operation

The System Tick 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 System Tick Timer is clocked from the CPU clock. In order to generate recurring interrupts at a specific interval, the LOAD register must be initialized with the correct value for the desired interval.

The block diagram of the System Tick Timer is shown below in the Figure 57.

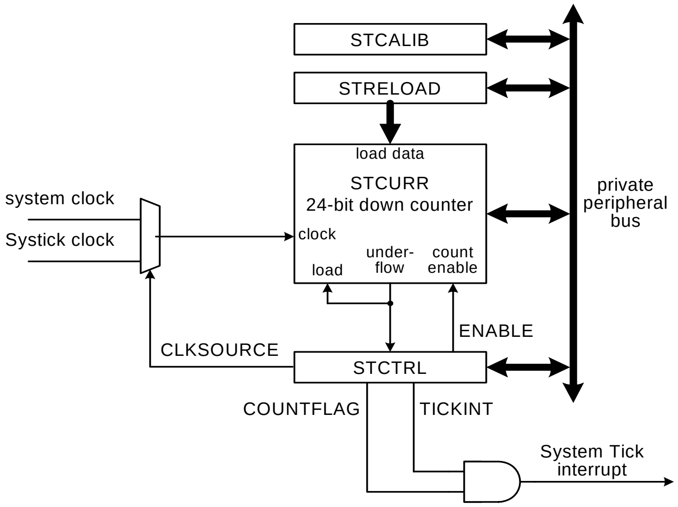


Fig 57. System tick timer block diagram

17.6 Register description

Table 284. Register overview: system tick timer (base address 0xE000 E000)

Name	Access	Address offset	Description	Reset value[1]
CTRL	R/W	0x010	System Timer Control and status register	0x0
LOAD	R/W	0x014	System Timer Reload value register	0x0
VAL	R/W	0x018	System Timer Current value register	0x0
CALIB	RO	0x01C	System Timer Calibration value register	0x0000 0004

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

17.6.1 System Timer Control and status register (CTRL - 0xE000 E010)

The CTRL register contains control information for the System Tick Timer, and provides a status flag.

Table 285. System Timer Control and status register (CTRL - 0xE000 E010) bit description

Bit	Symbol	Description	Reset value
0	ENABLE	System Tick counter enable. When 1, the counter is enabled. When 0, the counter is disabled.	0
1	TICKINT	System Tick interrupt enable. When 1, the System Tick interrupt is enabled. When 0, the System Tick interrupt is disabled. When enabled, the interrupt is generated when the System Tick counter counts down to 0.	0
2	CLKSOURCE	System Tick clock source selection. When 1, the system clock (CPU) clock is selected. When 0, the output clock from the system tick clock divider (SYSTICKDIV) is selected as the reference clock. In this case, the core clock must be at least 2.5 times faster than the reference clock otherwise the count values are unpredictable.	0
15:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
16	COUNTFLAG	System Tick counter flag. This flag is set when the System Tick counter counts down to 0, and is cleared by reading 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

17.6.2 System Timer Reload value register (LOAD - 0xE000 E014)

The LOAD register is set to the value that will be loaded into the System Tick Timer whenever it counts down to zero. This register is loaded by software as part of timer initialization. The CALIB register may be read and used as the value for LOAD if the CPU or external clock is running at the frequency intended for use with the CALIB value.

Table 286. System Timer Reload value register (LOAD - 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

17.6.3 System Timer Current value register (VAL - 0xE000 E018)

The VAL register returns the current count from the System Tick counter when it is read by software.

Table 287. System Timer Current value register (VAL - 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

17.6.4 System Timer Calibration value register (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 Table 43).

Table 288. System Timer Calibration value register (CALIB - 0xE000 E01C) bit description

Bit	Symbol	Value	Description	Reset value
23:0	TENMS		Factory preset	0x4
29:24	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
30	SKEW		Indicates whether the value of TENMS is precise. This can affect the suitability of SysTick as a software real time clock.	0
31	NOREF		Indicates whether a separate reference clock is available. This value is factory preset.	0

17.7 Example timer calculations

To use the system tick timer, do the following:

1. Program the LOAD register with the reload value RELOAD to obtain the desired time interval.
2. Clear the VAL register by writing to it. This ensures that the timer will count from the LOAD value rather than an arbitrary value when the timer is enabled.

The following examples illustrate selecting SysTick timer reload values for different system configurations. All of the examples calculate an interrupt interval of 10 milliseconds, as the SysTick timer is intended to be used, and there are no rounding errors.

System clock = 72 MHz

Program the CTRL register with the value 0x7 which selects the system clock as the clock source and enables the SysTick timer and the SysTick timer interrupt.

RELOAD = (system clock frequency × 10 ms) –1 = (72 MHz × 10 ms) –1 = 720000 –1 = 719999 = 0x000AFC7F

System tick timer clock = 24 MHz

Program the CTRL register with the value 0x3 which selects the clock from the system tick clock divider (use DIV = 3) as the clock source and enables the SysTick timer and the SysTick timer interrupt.

RELOAD = (system tick timer clock frequency × 10 ms) –1 = (24 MHz × 10 ms) –1 = 240000 –1 = 239999 = 0x0003A97F

System clock = 12 MHz

Program the CTRL register with the value 0x7 which selects the system clock as the clock source and enables the SysTick timer and the SysTick timer interrupt.

In this case the system clock is derived from the IRC clock.

RELOAD = (system clock frequency × 10 ms) –1 = (12 MHz × 10 ms) –1 = 120000 –1 = 119999 = 0x0001 D4BF

UM10375

Chapter 18: LPC13xx WatchDog Timer (WDT)

Rev. 5 — 21 June 2012 **User manual**

18.1 How to read this chapter

This chapter describes the WDT block without the windowed watchdog features and applies to the LPC1300 parts LPC1311/13/42/43.

18.2 Basic configuration

The WDT is configured using the following registers:

1. Pins: The WDT uses no external pins.
2. Power: In the SYSAHBCLKCTRL register, set bit 15 (Table 25).
3. Peripheral clock: Select the WDT clock source (Table 34) and enable the WDT peripheral clock by writing to the WDTCLKDIV register (Table 36).

   Remark: The frequency of the watchdog oscillator is undefined after reset. The watchdog oscillator frequency must be programmed by writing to the WDTOSCCTRL register (see Table 15) before using the watchdog oscillator for the WDT.
4. Lock features: Once the watchdog timer is enabled by setting the WDEN bit in the WDMOD register, the WDEN bit cannot be disabled.

18.3 Features

- Internally resets chip if not periodically reloaded.
- Debug mode.
- Enabled by software but requires a hardware reset or a Watchdog reset/interrupt to be disabled.
- Incorrect/Incomplete feed sequence causes reset/interrupt if enabled.
- Flag to indicate Watchdog reset.
- Programmable 24-bit timer with internal pre-scaler.
- Selectable time period from ($T_{WDCLK} \times 256 \times 4$) to ($T_{WDCLK} \times 2^{24} \times 4$) in multiples of $T_{WDCLK} \times 4$.
- The Watchdog clock (WDCLK) source is selected in the syscon block from the Internal RC oscillator (IRC), the main clock, or the Watchdog oscillator, see Table 15. 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.

18.4 Applications

The purpose of the Watchdog is to reset the microcontroller within a reasonable amount of time if it enters an erroneous state. When enabled, the Watchdog will generate a system reset if the user program fails to feed (or reload) the Watchdog within a predetermined amount of time.

18.5 Description

The Watchdog consists of a divide by 4 fixed pre-scaler and a 24-bit counter. The clock is fed to the timer via a pre-scaler. The timer decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum Watchdog interval is ($T_{WDCLK} \times 256 \times 4$) and the maximum Watchdog interval is ($T_{WDCLK} \times 2^{24} \times 4$) in multiples of ($T_{WDCLK} \times 4$). The Watchdog should be used in the following manner:

1. Set the Watchdog timer constant reload value in WDTC register.
2. Setup the Watchdog timer operating mode in WDMOD register.
3. Enable the Watchdog by writing 0xAA followed by 0x55 to the WDFEED register.
4. The Watchdog should be fed again before the Watchdog counter underflows to prevent reset/interrupt.

When the Watchdog is in the reset mode and the counter underflows, 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.

18.6 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. Several clocks can be used as a clock source for wdt_clk clock: the IRC, the watchdog oscillator, and the main clock. The clock source is selected in the syscon block (see Table 34). The WDCLK has its own clock divider (Table 36) which can also disable this clock.

There is some synchronization logic between these two clock domains. When the WDMOD and WDTC registers are updated by APB operations, the new value will take effect in 3 WDCLK cycles on the logic in the WDCLK clock domain. When the watchdog timer is counting on WDCLK, the synchronization logic will first lock the value of the counter on WDCLK and then synchronize it with the PCLK for reading as the WDTV register by the CPU.

The watchdog oscillator can be powered down in the PDRUNCFG register (Table 55) if it is not used. The clock to the watchdog register block (PCLK) can be disabled in the SYSAHBCLKCTRL register (Table 25) for power savings.

Remark: The frequency of the watchdog oscillator is undefined after reset. The watchdog oscillator frequency must be programmed by writing to the WDTOSCCTRL register (see Table 15) before using the watchdog oscillator for the WDT.

18.7 Register description

The Watchdog contains four registers as shown in Table 289 below.

Table 289. Register overview: Watchdog timer (base address 0x4000 4000)

Name	Access	Address offset	Description	Reset Value[1]
WDMOD	R/W	0x000	Watchdog mode register. This register contains the basic mode and status of the Watchdog Timer.	0
WDTC	R/W	0x004	Watchdog timer constant register. This register determines the time-out value.	0xFF
WDFEED	WO	0x008	Watchdog feed sequence register. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in WDTC.	NA
WDTV	RO	0x00C	Watchdog timer value register. This register reads out the current value of the Watchdog timer.	0xFF

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

18.7.1 Watchdog Mode register (WDMOD - 0x4000 0000)

The WDMOD register controls the operation of the Watchdog through the combination of WDEN and RESET bits. Note that a watchdog feed must be performed before any changes to the WDMOD register take effect.

Table 290. Watchdog Mode register (WDMOD - address 0x4000 4000) bit description

Bit	Symbol	Description	Reset Value
0	WDEN	WDEN Watchdog enable bit (Set Only). When one, the watchdog timer is running.	0
1	WDRESET	WDRESET Watchdog reset enable bit (Set Only). When 1, a watchdog time-out will cause a chip reset.	0
2	WDTOF	WDTOF Watchdog time-out flag. Set when the watchdog timer times out, cleared by software.	0 (After any reset except WDT)
3	WDINT	WDINT Watchdog interrupt flag (Read Only, not clearable by software).	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
31:8	-	reserved	-

Once the **WDEN** and/or **WDRESET** bits are set they can not be cleared by software. Both flags are cleared by reset or a Watchdog timer underflow.

WDTOF The Watchdog time-out flag is set when the Watchdog times out. This flag is cleared by software or any reset except the WDT reset.

WDINT The Watchdog interrupt flag is set when the Watchdog times out. This flag is cleared when any reset occurs. Once the watchdog interrupt is serviced, it can be disabled in the NVIC or the watchdog interrupt request will be generated indefinitely. the intent of the watchdog interrupt is to allow debugging watchdog activity without resetting the device when the watchdog overflows.

Watchdog reset or interrupt will occur any time the watchdog is running and has an operating clock source. Any clock source works in Sleep mode, and if a watchdog interrupt occurs in Sleep mode, it will wake up the device.

Table 291. 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: debug with the Watchdog interrupt but no WDRESET enabled. When this mode is selected, a watchdog counter underflow will set the WDINT flag and the Watchdog interrupt request will be generated.
1	1	Watchdog reset mode: operate with the Watchdog interrupt and WDRESET enabled. When this mode is selected, a watchdog counter underflow will reset the microcontroller. Although the Watchdog interrupt is also enabled in this case (WDEN = 1) it will not be recognized since the watchdog reset will clear the WDINT flag.

18.7.2 Watchdog Timer Constant register (WDTC - 0x4000 4004)

The WDTC register determines the time-out value. Every time a feed sequence occurs the WDTC content is reloaded in to the Watchdog timer. It’s a 32-bit register with 8 LSB set to 1 on reset. Writing values below 0xFF will cause 0x0000 00FF to be loaded to the WDTC. Thus the minimum time-out interval is $T_{WDCLK} \times 256 \times 4$.

Table 292. Watchdog Constant register (WDTC - address 0x4000 4004) bit description

Bit	Symbol	Description	Reset Value
23:0	COUNT	Watchdog time-out interval.	0x0000 00FF
31:24	-	Reserved	-

18.7.3 Watchdog Feed register (WDFEED - 0x4000 4008)

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. 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 293. Watchdog Feed register (WDFEED - address 0x4000 4008) bit description

Bit	Symbol	Description	Reset Value
7:0	FEED	Feed value should be 0xAA followed by 0x55.	-
31:8	-	reserved	-

18.7.4 Watchdog Timer Value register (WDTV - 0x4000 400C)

The WDTV register is used to read the current value of Watchdog timer.

When reading the value of the 24-bit timer, 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 294. Watchdog Timer Value register (WDTV - address 0x4000 000C) bit description

Bit	Symbol	Description	Reset Value
23:0	COUNT	Counter timer value.	0x0000 00FF
31:24	-	Reserved	-

18.8 Block diagram

The block diagram of the Watchdog is shown below in the Figure 58. The synchronization logic (PCLK - WDCLK) is not shown in the block diagram.

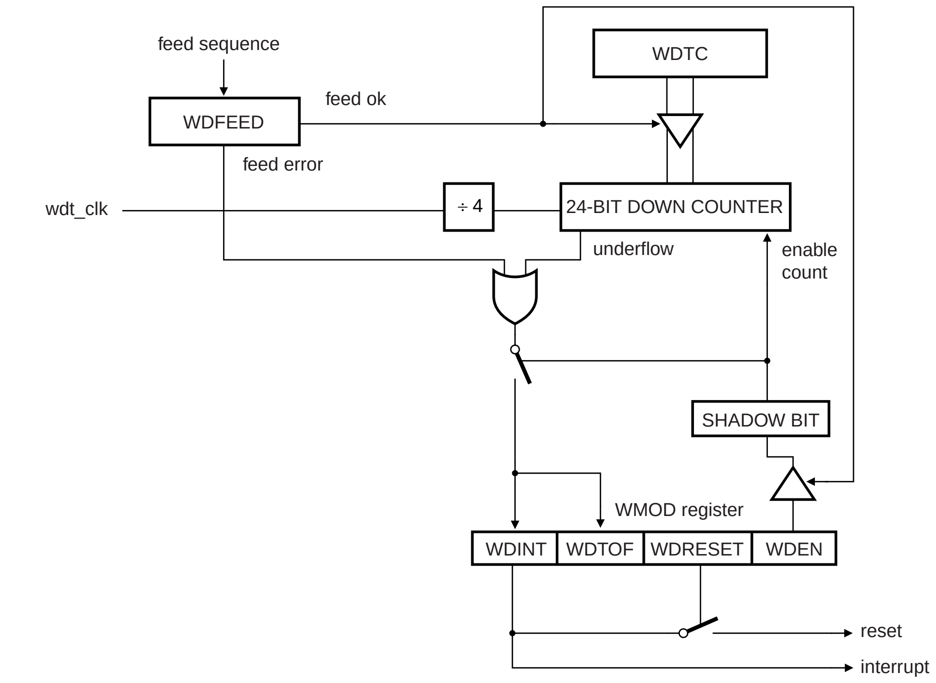


Fig 58. Watchdog block diagram

UM10375

Chapter 19: LPC13xx Windowed WatchDog Timer (WWDT)

Rev. 5 — 21 June 2012 User manual

19.1 How to read this chapter

This chapter describes the Windowed WDT available on the LPC1300L parts LPC1311/01 and LPC1313/01.

19.2 Basic configuration

The WDT is configured using the following registers:

1. Pins: The WDT uses no external pins.
2. Power: In the SYSAHBCLKCTRL register, set bit 15 (Table 25).
3. Peripheral clock: Select the WDT clock source (Table 34) and enable the WDT peripheral clock by writing to the WDTCLKDIV register (Table 36).

   Remark: The frequency of the watchdog oscillator is undefined after reset. The watchdog oscillator frequency must be programmed by writing to the WDTOSCCTRL register (see Table 15) before using the watchdog oscillator for the WDT.
4. Lock features: Once the watchdog timer is enabled by setting the WDEN bit in the WDMOD register, the following lock features are in effect:
   a. The WDEN cannot be disabled.
   b. The watch dog clock source cannot be changed. If the WDT is needed in Deep-sleep mode, select the watch dog oscillator as the clock source before setting the WDEN bit.

19.3 Features

- Internally resets chip if not reloaded during the programmable time-out period.
- Optional windowed operation requires reload to occur between a minimum and maximum time-out period, both programmable.
- Optional warning interrupt can be generated at a programmable time prior to watchdog time-out.
- Programmable 24-bit timer with internal fixed pre-scaler.
- Selectable time period from 1,024 watchdog clocks ($T_{WDCLK} \times 256 \times 4$) to over 67 million watchdog clocks ($T_{WDCLK} \times 2^{24} \times 4$) in increments of 4 watchdog clocks.
- “Safe” watchdog operation. Once enabled, requires a hardware reset or a Watchdog reset to be disabled.
- A dedicated on-chip watchdog oscillator provides a reliable clock source that cannot be turned off when the Watchdog Timer is running.
- Incorrect feed sequence causes immediate watchdog reset if the watchdog is enabled.
- The watchdog reload value can optionally be protected such that it can only be changed after the “warning interrupt” time is reached.
- Flag to indicate Watchdog reset.

19.4 Applications

The purpose of the Watchdog Timer is to reset the microcontroller within a reasonable amount of time if it 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.

.

19.5 General description

The Watchdog consists of a fixed divide-by-4 pre-scaler and a 24-bit counter which decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum Watchdog interval is ($T_{WDCLK} \times 256 \times 4$) and the maximum Watchdog interval is ($T_{WDCLK} \times 2^{24} \times 4$) in multiples of ($T_{WDCLK} \times 4$). The Watchdog should be used in the following manner:

- Set the Watchdog timer constant reload value in WDTC register.
- Setup the Watchdog timer operating mode in WDMOD register.
- Set a value for the watchdog window time in WDWINDOW register if windowed operation is required.
- Set a value for the watchdog warning interrupt in the WDWARNINT register if a warning interrupt is required.
- Enable the Watchdog by writing 0xAA followed by 0x55 to the WDFEED 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 WDWARNINT register.

The block diagram of the Watchdog is shown below in the Figure 59. The synchronization logic (PCLK - WDCLK) is not shown in the block diagram.

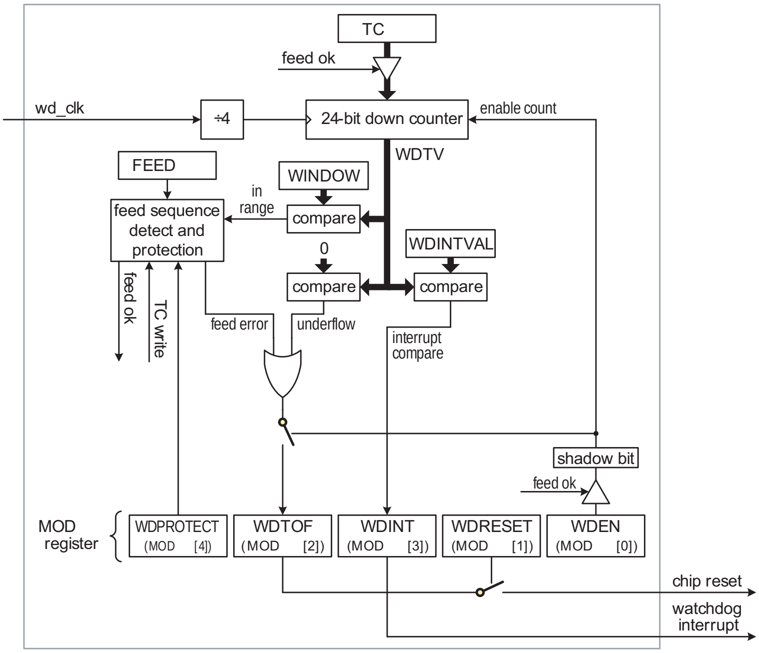


Fig 59. Watchdog Timer block diagram

19.6 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. Several clocks can be used as a clock source for wdt_clk clock: the IRC, the watchdog oscillator, and the main clock. The clock source is selected in the syscon block (see Table 34). The WDCLK has its own clock divider (Table 36) which can also disable this clock.

There is some synchronization logic between these two clock domains. When the WDMOD and WDTC registers are updated by APB operations, the new value will take effect in 3 WDCLK cycles on the logic in the WDCLK clock domain. When the watchdog timer is counting on WDCLK, the synchronization logic will first lock the value of the counter on WDCLK and then synchronize it with the PCLK for reading as the WDTV register by the CPU.

The watchdog oscillator can be powered down in the PDRUNCFG register (Table 55) if it is not used. The clock to the watchdog register block (PCLK) can be disabled in the SYSAHBCLKCTRL register (Table 25) for power savings.

Remark: The frequency of the watchdog oscillator is undefined after reset. The watchdog oscillator frequency must be programmed by writing to the WDTOSCCTRL register (see Table 15) before using the watchdog oscillator for the WDT.

19.7 Register description

The Watchdog contains four registers as shown in Table 295 below.

Table 295. Register overview: Watchdog timer (base address 0x4000 4000)

Name	Access	Address offset	Description	Reset value[1]
WDMOD	R/W	0x000	Watchdog mode register. This register contains the basic mode and status of the Watchdog Timer.	0
WDTC	R/W	0x004	Watchdog timer constant register. This register determines the time-out value.	0xFF
WDFEED	WO	0x008	Watchdog feed sequence register. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in WDTC.	-
WDTV	RO	0x00C	Watchdog timer value register. This register reads out the current value of the Watchdog timer.	0xFF
WDWARNINT	R/W	0x014	Watchdog Warning Interrupt compare value.	0
WDWINDOW	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.

19.7.1 Watchdog Mode register

The WDMOD register controls the operation of the Watchdog as per the combination of WDEN and RESET bits. Note that a watchdog feed must be performed before any changes to the WDMOD register take effect.

Table 296: Watchdog Mode register (WDMOD - 0x4000 4000) bit description

Bit	Symbol	Value	Description	Reset value
0	WDEN		Watchdog enable bit. This bit is Set Only. **Remark:** Setting this bit to one also locks the watchdog clock source. Once the watchdog timer is enabled, the watchdog timer clock source cannot be changed. If the watchdog timer is needed in Deep-sleep mode, the watchdog clock source must be changed to the watchdog oscillator before setting this bit to one.	0
		0	The watchdog timer is stopped.	
		1	The watchdog timer is running.	
1	WDRESET		Watchdog reset enable bit. This bit is Set Only.	0
		0	A watchdog timeout will not cause a chip reset.	
		1	A watchdog timeout will cause a chip reset.	
2	WDTOF		Watchdog time-out flag. Set when the watchdog timer times out, by a feed error, or by events associated with WDPROTECT, cleared by software. Causes a chip reset if WDRESET = 1.	0 (Only after external reset)
3	WDINT		Watchdog interrupt flag. Set when the timer reaches the value in WDWARNINT. Cleared by software.	0

Table 296: Watchdog Mode register (WDMOD - 0x4000 4000) bit description

Bit	Symbol	Value	Description	Reset value
4	WDPROTECT		Watchdog update mode. This bit is Set Only.	0
		0	The watchdog reload value (WDTC) can be changed at any time.	
		1	The watchdog reload value (WDTC) can be changed only after the counter is below the value of WDWARNINT and WDWINDOW. **Note**: this mode is intended for use only when WDRESET =1.	
31:5	-		Reserved. Read value is undefined, only zero should be written.	-

Once the **WDEN**, **WDPROTECT**, or **WDRESET** bits are set they can not be cleared by software. Both flags are cleared by an external reset or a Watchdog timer reset.

WDTOF The Watchdog time-out flag is set when the Watchdog times out, when a feed error occurs, or when 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.

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.

Watchdog reset or interrupt will occur any time the watchdog is running. If a watchdog interrupt occurs in Sleep mode, it will wake up the device.

Table 297. Watchdog operating modes selection

WDEN	WDRESET	Mode of Operation
0	X (0 or 1)	Debug/Operate without the Watchdog running.
1	0	Watchdog interrupt mode: the watchdog warning interrupt will be generated but watchdog reset will not. When this mode is selected, the watchdog counter reaching the value specified by WDWARNINT will set the WDINT flag and the Watchdog interrupt request will be generated.
1	1	Watchdog reset mode: both the watchdog interrupt and watchdog reset are enabled. When this mode is selected, the watchdog counter reaching the value specified by WDWARNINT will set the WDINT flag and the Watchdog interrupt request will be generated, and the watchdog counter reaching zero will reset the microcontroller. A watchdog feed prior to reaching the value of WDWINDOW will also cause a watchdog reset.

19.7.2 Watchdog Timer Constant register

The WDTC register determines the time-out value. Every time a feed sequence occurs the WDTC content is reloaded in to the Watchdog timer. This is pre-loaded with the value 0x00 00FF upon reset. Writing values below 0xFF will cause 0x00 00FF to be loaded into the WDTC. Thus the minimum time-out interval is $T_{WDCLK} \times 256 \times 4$.

If the WDPROTECT bit in WDMOD = 1, an attempt to change the value of WDTC before the watchdog counter is below the values of WDWARNINT and WDWINDOW will cause a watchdog reset and set the WDTOF flag.

Table 298: Watchdog Timer Constant register (WDTC - 0x4000 4004) bit description

Bit	Symbol	Description	Reset value
23:0	Count	Watchdog time-out interval.	0x00 00FF
31:24	-	Reserved. Read value is undefined, only zero should be written.	NA

19.7.3 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.

Table 299: Watchdog Feed register (WDFEED - 0x4000 4008) bit description

Bit	Symbol	Description	Reset value
7:0	Feed	Feed value should be 0xAA followed by 0x55.	-
31:8	-	Reserved	-

19.7.4 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 300: Watchdog Timer Value register (WDTV - 0x4000 400C) bit description

Bit	Symbol	Description	Reset value
23:0	Count	Counter timer value.	0x00 00FF
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

19.7.5 Watchdog Timer Warning Interrupt register

The WDWARNINT register determines the watchdog timer counter value that will generate a watchdog interrupt. When the watchdog timer counter matches the value defined by 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 301: Watchdog Timer Warning Interrupt register (WDWARNINT - 0x4000 4014) bit description

Bit	Symbol	Description	Reset value
9:0	WARNINT	Watchdog warning interrupt compare value.	0
31:10	-	Reserved. Read value is undefined, only zero should be written.	-

19.7.6 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.

Table 302: Watchdog Timer Window register (WDWINDOW - 0x4000 4018) bit description

Bit	Symbol	Description	Reset value
23:0	WINDOW	Watchdog window value.	0xFF FFFF
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

19.7.7 Watchdog timing examples

The following figures illustrate several aspects of Watchdog Timer operation.

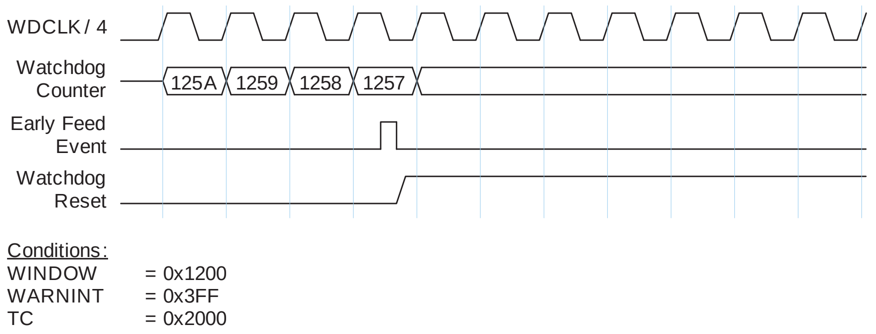
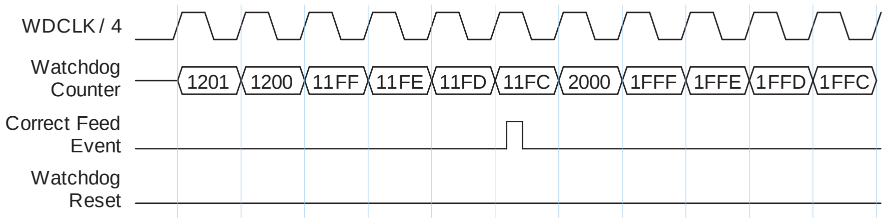
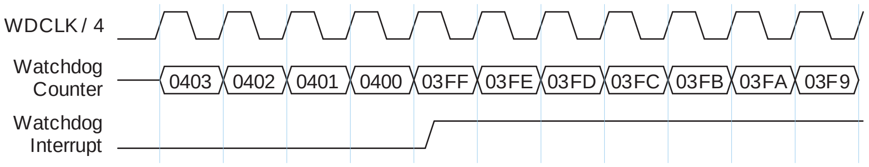


Fig 60. Early Watchdog Feed with Windowed Mode Enabled



Conditions:
WDWINDOW = 0x1200
WDWARNINT = 0x3FF
WDTC = 0x2000

Fig 61. Correct Watchdog Feed with Windowed Mode Enabled



Conditions:
WINDOW = 0x1200
WARNINT = 0x3FF
TC = 0x2000

Fig 62. Watchdog Warning Interrupt

UM10375

Chapter 20: LPC13xx Analog-to-Digital Converter (ADC)

Rev. 5 — 21 June 2012 User manual

20.1 How to read this chapter

The ADC block is identical for all LPC13xx parts.

20.2 Basic configuration

The ADC is configured using the following registers:

1. Pins: The ADC pin functions are configured in the IOCONFIG register block (Section 7.4).
2. Power and peripheral clock: In the SYSAHBCLKCTRL register, set bit 13 (Table 25). Power to the ADC at run-time is controlled through the PDRUNCFG register (Table 55).

20.3 Features

- 10-bit successive approximation Analog-to-Digital Converter (ADC).
- Input multiplexing among 8 pins.
- Power-down mode.
- Measurement range 0 to 3.6 V. Do not exceed the V_{DD} voltage level.
- 10-bit conversion time ≥ 2.44 μs.
- 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.

20.4 Pin description

Table 303 gives a brief summary of the ADC related pins.

Table 303. 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. **Remark:** While the pins are 5 V tolerant in digital mode, the maximum input voltage must not exceed V_{DD} when the pins are configured as analog inputs.
V_{DD}	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.

20.5 Clocking and power control

The peripheral clock to the ADC (PCLK) is provided by the system clock (see Figure 3). This clock can be disabled through bit 13 in the SYSAHBCLKCTRL register (Table 25) for power savings.

The ADC can be powered down at run-time using the PDRUNCFG register (Table 55).

Basic clocking for the A/D converters is determined by the peripheral ADC clock PCLK. A programmable divider is included in each converter to scale this clock to the 4.5 MHz (max) clock needed by the successive approximation process. An accurate conversion requires 11 clock cycles.

20.6 Register description

The ADC contains registers organized as shown in Table 304.

Table 304. Register overview: ADC (base address 0x4001 C000)

Name	Access	Address offset	Description	Reset Value[1]
AD0CR	R/W	0x000	A/D Control Register. The AD0CR register must be written to select the operating mode before A/D conversion can occur.	0x0000 0000
AD0GDR	R/W	0x004	A/D Global Data Register. Contains the result of the most recent A/D conversion.	NA
-	-	0x008	Reserved.	-
AD0INTEN	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
AD0DR0	R/W	0x010	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0	NA
AD0DR1	R/W	0x014	A/D Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1.	NA
AD0DR2	R/W	0x018	A/D Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2.	NA
AD0DR3	R/W	0x01C	A/D Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3.	NA
AD0DR4	R/W	0x020	A/D Channel 4 Data Register. This register contains the result of the most recent conversion completed on channel 4.	NA
AD0DR5	R/W	0x024	A/D Channel 5 Data Register. This register contains the result of the most recent conversion completed on channel 5.	NA
AD0DR6	R/W	0x028	A/D Channel 6 Data Register. This register contains the result of the most recent conversion completed on channel 6.	NA
AD0DR7	R/W	0x02C	A/D Channel 7 Data Register. This register contains the result of the most recent conversion completed on channel 7.	NA
AD0STAT	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

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

20.6.1 A/D Control Register (AD0CR - 0x4001 C000)

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 305: A/D Control Register (AD0CR - address 0x4001 C000) 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. Bit 0 selects Pin AD0, bit 1 selects pin AD1,..., and bit 7 selects pin AD7. In software-controlled mode (BURST = 0), only one channel can be selected, i.e. only one of these bits should be 1. In hardware scan mode (BURST = 1), any numbers of channels can be selected, i.e any or all bits can be set to 1. If all bits are set to 0, channel 0 is selected automatically (SEL = 0x01).	0x00
15:8	CLKDIV		The APB clock (PCLK) is divided by CLKDIV +1 to produce the clock for the ADC, which should be less than or equal to 4.5 MHz. Typically, software should program the smallest value in this field that yields a clock of 4.5 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 select	0
		0	Software-controlled mode: Conversions are software-controlled and require 11 clocks.	
		1	Hardware scan mode: The AD converter does repeated conversions at the rate selected by the CLKS field, scanning (if necessary) through the pins selected by 1s in the SEL field. The first conversion after the start corresponds to the least-significant bit set to 1 in the SEL field, then the next higher bits (pins) set to 1 are scanned if applicable. Repeated conversions can be terminated by clearing this bit, but the conversion in progress when this bit is cleared will be completed. **Important:** START bits must be 000 when BURST = 1 or conversions will not start.	
19:17	CLKS		This field selects the number of clocks used for each conversion in Burst mode, and the number of bits of accuracy of the result in the LS bits of ADDR, between 11 clocks (10 bits) and 4 clocks (3 bits).	000
		0x0	11 clocks / 10 bits	
		0x1	10 clocks / 9 bits	
		0x2	9 clocks / 8 bits	
		0x3	8 clocks / 7 bits	
		0x4	7 clocks / 6 bits	
		0x5	6 clocks / 5 bits	
		0x6	5 clocks / 4 bits	
		0x7	4 clocks / 3 bits	
23:20	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
26:24	START		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.	

Table 305: A/D Control Register (AD0CR - address 0x4001 C000) bit description

Bit	Symbol	Value	Description	Reset Value
		0x4	Start conversion when the edge selected by bit 27 occurs on CT32B0_MAT0. Timer match function does not need to be selected on the device pin.	
		0x5	Start conversion when the edge selected by bit 27 occurs on CT32B0_MAT1. Timer match function does not need to be selected on the device pin.	
		0x6	Start conversion when the edge selected by bit 27 occurs on CT16B0_MAT0. Timer match function does not need to be selected on the device pin.	
		0x7	Start conversion when the edge selected by bit 27 occurs on CT16B0_MAT1. Timer match function does not need to be selected on the device pin.	
27	EDGE		This bit is significant only when the START field contains 010-111. In these cases:	0
		0	Start conversion on a rising edge on the selected CAP/MAT signal.	
		1	Start conversion on a falling edge on the selected CAP/MAT signal.	
31:28	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

20.6.2 A/D Global Data Register (AD0GDR - 0x4001 C004)

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 306: A/D Global Data Register (AD0GDR - address 0x4001 C004) bit description

Bit	Symbol	Description	Reset Value
5:0	-	Reserved.	0
15:6	V_VREF	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} pin. 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.	0
26:24	CHN	These bits contain the channel from which the V_VREF bits were converted.	X
29:27	-	Reserved.	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 V_VREF 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

20.6.3 A/D Interrupt Enable Register (AD0INTEN - 0x4001 C00C)

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 307: A/D Interrupt Enable Register (AD0INTEN - address 0x4001 C00C) 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.	0

20.6.4 A/D Data Registers (AD0DR0 to AD0DR7 - 0x4001 C010 to 0x4001 C02C)

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 308: A/D Data Registers (AD0DR0 to AD0DR7 - addresses 0x4001 C010 to 0x4001 C02C) bit description

Bit	Symbol	Description	Reset Value
5:0	-	Reserved.	0
15:6	V_VREF	When DONE is 1, this field contains a binary fraction representing the voltage on the ADn pin, divided by the voltage on the V_{REF} pin. Zero in the field indicates that the voltage on the ADn pin was less than, equal to, or close to that on V_{REF}, while 0x3FF indicates that the voltage on AD input was close to, equal to, or greater than that on V_{REF}.	NA
29:16	-	Reserved.	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 V_VREF 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

20.6.5 A/D Status Register (AD0STAT - 0x4001 C030)

The A/D Status register allows checking the status of all A/D channels simultaneously. The DONE and OVERRUN flags appearing in the AD0DRn 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 309: A/D Status Register (AD0STAT - address 0x4001 C030) 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.	0

20.7 Operation

20.7.1 Hardware-triggered conversion

If the BURST bit in the ADCR0 is 0 and the START field contains 010-111, the A/D converter will start a conversion when a transition occurs on a selected pin or timer match signal.

20.7.2 Interrupts

An interrupt is requested to the interrupt controller when the ADINT bit in the ADSTAT register is 1. The ADINT bit is one when any of the DONE bits of A/D channels that are enabled for interrupts (via the ADINTEN register) are one. Software can use the Interrupt Enable bit in the interrupt controller that corresponds to the ADC to control whether this results in an interrupt. The result register for an A/D channel that is generating an interrupt must be read in order to clear the corresponding DONE flag.

UM10375

Chapter 21: LPC13xx Flash memory programming firmware

Rev. 5 — 21 June 2012 **User manual**

21.1 How to read this chapter

See Table 310 for different flash configurations and functionality. The LPC131x parts do not have an USB interface and only the UART ISP option is supported.

Table 310. LPC13xx flash configurations

Type number	Flash	ISP via USB	ISP via UART
LPC1311FHN33	8 kB	no	yes
LPC1313FBD48	32 kB	no	yes
LPC1313FHN33	32 kB	no	yes
LPC1342FHN33	16 kB	yes	yes
LPC1342FBD48	16 kB	yes	yes
LPC1343FBD48	32 kB	yes	yes
LPC1343FHN33	32 kB	yes	yes
LPC1311FHN33/01	8 kB	no	yes
LPC1313FHN33/01	32 kB	no	yes
LPC1313FBD48/01	32 kB	no	yes

Remark: In addition to the ISP and IAP commands, a register can be accessed in the flash controller block to configure flash memory access times, see Section 21.16.1.

21.2 Bootloader

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

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

The bootloader version can be read by ISP/IAP calls (see Section 21.13.12 or Section 21.14.6) and is part of the chip marking for some LPC13xx parts (see Table 311).

Table 311. Bootloader versions

Part	Bootloader version read by ISP/IAP	Top-side marking[1]	Notes
LPC1311	5.1	no marking	-
LPC1313	5.1	no marking	-
LPC1342	5.2	1	See Section 21.2.1.
LPC1343	5.2	1	See Section 21.2.1.

[1] Typical LPC134x devices have the following top-side marking:
LPC1343x

xxxxxxx
xxYYWW<bootloader version>R[x]

21.2.1 Bootloader code version 5.2 notes

In bootloader version 5.2 (LPC134x parts), the mass storage device state machine uses an uninitialized variable. This has two consequences:

1. In the user code, the memory location must be initialized as follows to create a work-around for this issue:

```
*((unit32_t *)(0x1000 0054)) = 0x0;
```

2. If the USB ISP mode is entered on power-up (see Section 21.3), the memory is not initialized, and no user code is executed which could write to this memory location. Therefore the device times out when first connected to the Windows operating system, and the MSD disk only appears after a time-out and retry, which takes 45 sec or longer. A work-around for the time-out issue is not available.

21.3 Features

- In-System Programming: In-System programming (ISP) is programming or reprogramming the on-chip flash memory, using the bootloader software and UART serial port or the USB interface. 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.
- The LPC134x supports ISP from the USB port through enumeration as a Mass Storage Class (MSC) Device when connected to a USB host interface (Windows operating system only).
- Flash access times can be configured through a register in the flash controller block.
- Erase time for one sector is 100 ms ± 5%. Programming time for one block of 256 bytes is 1 ms ± 5%.

21.4 Description

The bootloader code is executed every time the part is powered on or reset (see Figure 63). The loader can either execute the ISP command handler or the user application code, or it can obtain the boot image as an attached MSC device through USB. A LOW level during reset at the PIO0_1 pin is considered an external hardware request to start the ISP command handler or the USB device enumeration without checking for a valid user code first. The state of PIO0_3 determines whether the UART or USB interface will be used:

- If PIO0_3 is sampled HIGH, the bootloader connects the LPC134x as a MSC USB device to a PC host. The LPC134x flash memory space is represented as a drive in the host's Windows operating system.
- If PIO0_3 is sampled LOW, the bootloader configures the UART serial port and calls the ISP command handler.

Remark: On the LPC131x parts (no USB), the state of pin PIO0_3 does not matter.

Assuming that power supply pins are at their nominal levels when the rising edge on RESET pin is generated, it may take up to 3 ms before PIO0_1 is sampled and the decision whether to continue with user code or ISP handler/USB is made. If PIO0_1 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_1 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.

Remark: The sampling of pin PIO0_1 can be disabled through programming flash location 0x0000 02FC (see Section 21.12.1).

21.5 Memory map after any reset

The boot block is 16 kB in size and is located in the memory region starting from the address 0x1FFF 0000. The bootloader is designed to run from this memory area, but both the ISP and IAP software use parts of the on-chip RAM. The RAM usage is described 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.

21.6 Flash content protection mechanism

The LPC13xx is equipped with the Error Correction Code (ECC) capable Flash memory. The purpose of an error correction module is twofold. Firstly, it decodes data words read from the memory into output data words. Secondly, it encodes data words to be written to the memory. The error correction capability consists of single bit error correction with Hamming code.

The operation of ECC is transparent to the running application. The ECC content itself is stored in a flash memory not accessible by user’s code to either read from it or write into it on its own. A byte of ECC corresponds to every consecutive 128 bits of the user accessible Flash. Consequently, Flash bytes from 0x0000 0000 to 0x0000 000F are protected by the first ECC byte, Flash bytes from 0x0000 0010 to 0x0000 001F are protected by the second ECC byte, etc.

Whenever the CPU requests a read from user’s Flash, both 128 bits of raw data containing the specified memory location and the matching ECC byte are evaluated. If the ECC mechanism detects a single error in the fetched data, a correction will be applied before data are provided to the CPU. When a write request into the user’s Flash is made, write of user specified content is accompanied by a matching ECC value calculated and stored in the ECC memory.

When a sector of Flash memory is erased, the corresponding ECC bytes are also erased. Once an ECC byte is written, it can not be updated unless it is erased first. Therefore, for the implemented ECC mechanism to perform properly, data must be written into the flash memory in groups of 16 bytes (or multiples of 16), aligned as described above.

21.7 Criterion for Valid User Code

The reserved ARM Cortex-M3 exception vector location 7 (offset 0x0000 001C in the vector table) should contain the 2’s complement of the check-sum of table entries 0 through 6. This causes the checksum of the first 8 table entries to be 0. The bootloader code checksums the first 8 locations in sector 0 of the flash. If the result is 0, then execution control is transferred to the user code.

If the signature is not valid, the auto-baud routine synchronizes with the host via the serial port (UART) or boots from the USB port (PIO0_3 is sampled HIGH).

If the UART is selected, the host should send a ’?’ (0x3F) as a synchronization character and wait for a response. The host side serial port settings should be 8 data bits, 1 stop bit and no parity. The auto-baud routine measures the bit time of the received synchronization character in terms of its own frequency and programs the baud rate generator of the serial port. It also sends an ASCII string ("Synchronized<CR><LF>") to the host. In response to this 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. In UART ISP mode, the LPC13xx is clocked by the IRC and the crystal frequency is ignored.

For more details on Reset, PLL and startup/boot code interaction see Section 3.6.

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 21.13 “ISP commands” on page 330.

21.8 ISP/IAP 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.

21.8.1 ISP command format

"Command Parameter_0 Parameter_1 ... Parameter_n<CR><LF>" "Data" (Data only for Write commands).

21.8.2 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).

21.8.3 ISP data format

The data stream is in UU-encoded 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.

21.8.4 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.

21.8.5 ISP command abort

Commands can be aborted by sending the ASCII control character "ESC". This feature is not documented as a command under "ISP Commands" section. Once the escape code is received the ISP command handler waits for a new command.

21.8.6 Interrupts during ISP

The boot block interrupt vectors located in the boot block of the flash are active after any reset.

21.8.7 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. Before making any IAP call, either disable interrupts or ensure that user interrupt vectors are active in RAM and that the interrupt handlers reside in RAM. The IAP code does not use or disable interrupts.

21.8.8 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 bytes. The maximum stack usage is 256 bytes and grows downwards.

21.8.9 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 grows downwards.

21.9 USB communication protocol

The LPC134x is enumerated as a Mass Storage Class (MSC) device to a PC or another embedded system. In order to connect via the USB interface, the LPC134x must use the external crystal at a frequency of 12 MHz. The MSC device presents an easy integration with the PC’s Windows operating system. The LPC134x flash memory space is represented as a drive in the host file system. The entire available user flash is mapped to a file of the size of the LPC134x flash in the host’s folder with the default name ‘firmware.bin’. The ‘firmware.bin’ file can be deleted and a new file can be copied into the directory, thereby updating the user code in flash. Note that the filename of the new flash image file is not important. After a reset or a power cycle, the new file is visible in the host’s file system under it’s default name ‘firmware.bin’.

Remark: USB ISP commands are supported for Windows operating system only.

The code read protection (CRP, see Table 312) level determines how the flash is reprogrammed:

If CRP1 or CRP2 is enabled, the user flash is erased when the file is deleted.

If CRP1 is enabled or no CRP is selected, the user flash is erased and reprogrammed when the new file is copied. However, only the area occupied by the new file is erased and reprogrammed.

Remark: The only Windows commands supported for the LPC134x flash image folder are copy and delete.

Three Code Read Protection (CRP) levels can be enabled for flash images updated through USB (see Section 21.12 for details). The volume label on the MSCD indicates the CRP status.

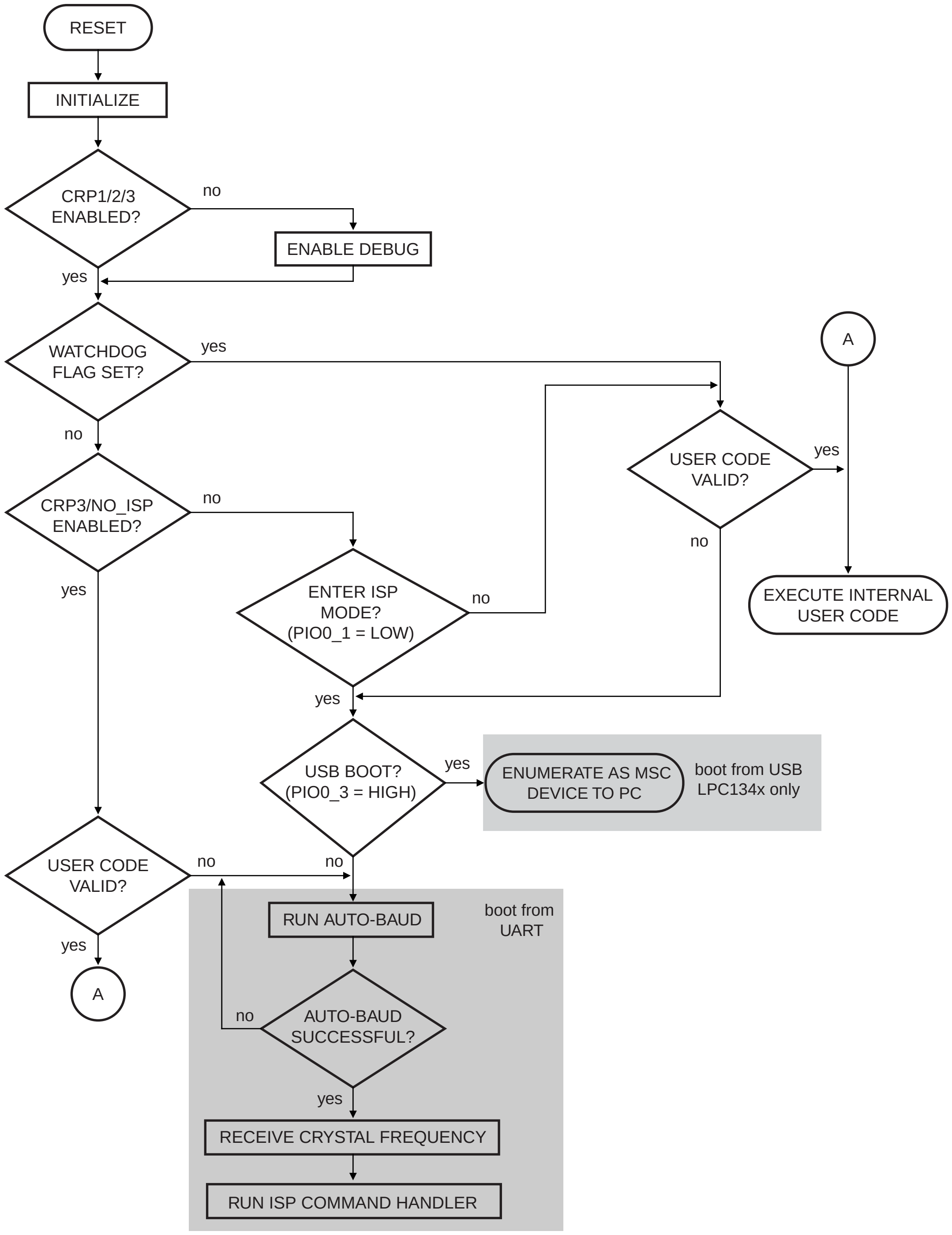
Table 312. CRP levels for USB boot images

CRP status	Volume label	Description
No CRP	CRP DISABLD	The user flash can be read or written.
CRP1	CRP1 ENABLD	The user flash content cannot be read but can be updated. The flash memory sectors are updated depending on the new firmware image.
CRP2	CRP2 ENABLD	The user flash content cannot be read but can be updated. The entire user flash memory is erased before writing the new firmware image.
CRP3	CRP3 ENABLD	The user flash content cannot be read or updated. The bootloader always executes the user application if valid.

21.9.1 Usage note

When programming flash images via Flash Magic or Serial Wire Debugger (SWD), the user code valid signature is automatically inserted by the programming utility. When using USB ISP, the user code valid signature must be either part of the vector table, or the axf or binary file must be post-processed to insert the checksum.

21.10 Boot process flowchart



(1) For details on handling the crystal frequency, see Section 21.14.8.

(2) For details on available ISP commands based on the CRP settings, see Section 21.12.

Fig 63. Boot process flowchart

21.11 Sector numbers

Some IAP and ISP commands operate on sectors and specify sector numbers. The following table shows the correspondence between sector numbers and memory addresses for LPC13xx devices.

Table 313. LPC13xx flash sectors

Sector number	Sector size [kB]	Address range	LPC1311	LPC1313	LPC1342	LPC1343
0	4	0x0000 0000 - 0x0000 0FFF	yes	yes	yes	yes
1	4	0x0000 1000 - 0x0000 1FFF	yes	yes	yes	yes
2	4	0x0000 2000 - 0x0000 2FFF	-	yes	yes	yes
3	4	0x0000 3000 - 0x0000 3FFF	-	yes	yes	yes
4	4	0x0000 4000 - 0x0000 4FFF	-	yes	-	yes
5	4	0x0000 5000 - 0x0000 5FFF	-	yes	-	yes
6	4	0x0000 6000 - 0x0000 6FFF	-	yes	-	yes
7	4	0x0000 7000 - 0x0000 7FFF	-	yes	-	yes

21.12 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 314. Code Read Protection (CRP) options

Name	Pattern programmed in 0x0000 02FC	Description
NO_ISP	0x4E69 7370	Prevents sampling of pin PIO0_1 for entering ISP mode. PIO0_1 is available for other uses.
CRP1	0x12345678	Access to chip via the SWD pins is disabled. This mode allows partial flash update using the following ISP commands and restrictions: • Write to RAM command 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_1 LOW is disabled if a valid user code is present in flash sector 0. This mode effectively disables ISP override using PIO0_1 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 315. Code Read Protection hardware/software interaction

CRP option	User Code Valid	PIO0_1 pin at reset	SWD enabled	LPC13xx enters ISP mode	partial flash Update in ISP mode
None	No	x	Yes	Yes	Yes
None	Yes	High	Yes	No	NA
None	Yes	Low	Yes	Yes	Yes
CRP1	Yes	High	No	No	NA
CRP1	Yes	Low	No	Yes	Yes

Table 315. Code Read Protection hardware/software interaction *…continued*

CRP option	User Code Valid	PIO0_1 pin at reset	SWD enabled	LPC13xx enters ISP mode	partial flash Update in ISP mode
CRP2	Yes	High	No	No	NA
CRP2	Yes	Low	No	Yes	No
CRP3	Yes	x	No	No	NA
CRP1	No	x	No	Yes	Yes
CRP2	No	x	No	Yes	No
CRP3	No	x	No	Yes	No

Table 316. ISP commands allowed for different CRP levels

ISP command	CRP1	CRP2	CRP3 (no entry in ISP mode allowed)
Unlock	yes	yes	n/a
Set Baud Rate	yes	yes	n/a
Echo	yes	yes	n/a
Write to RAM	yes; above 0x1000 0300 only	no	n/a
Read Memory	no	no	n/a
Prepare sector(s) for write operation	yes	yes	n/a
Copy RAM to flash	yes; not to sector 0	no	n/a
Go	no	no	n/a
Erase sector(s)	yes; sector 0 can only be erased when all sectors are erased.	yes; all sectors only	n/a
Blank check sector(s)	no	no	n/a
Read Part ID	yes	yes	n/a
Read Boot code version	yes	yes	n/a
Compare	no	no	n/a
ReadUID	yes	yes	n/a

In case a CRP mode is enabled and access to the chip is allowed via the ISP, an unsupported or restricted ISP command will be terminated with return code CODE_READ_PROTECTION_ENABLED.

21.12.1 ISP entry protection

In addition to the three CRP modes, the user can prevent the sampling of pin PIO0_1 for entering ISP mode and thereby release pin PIO0_1 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.

The NO_ISP mode is identical to the CRP3 mode except for SWD access, which is allowed in NO_ISP mode but disabled in CRP3 mode. The NO_ISP mode does not offer any code protection.

21.13 ISP commands

The following commands are accepted by the ISP command handler. Detailed status codes are supported for each command. The command handler sends the return code INVALID_COMMAND when an undefined command is received. Commands and return codes are in ASCII format.

CMD_SUCCESS is sent by ISP command handler only when received ISP command has been completely executed and the new ISP command can be given by the host. Exceptions from this rule are "Set Baud Rate", "Write to RAM", "Read Memory", and "Go" commands.

Table 317. ISP command summary

ISP Command	Usage	Described in
Unlock	U <Unlock Code>	Table 318
Set Baud Rate	B <Baud Rate> <stop bit>	Table 319
Echo	A <setting>	Table 320
Write to RAM	W <start address> <number of bytes>	Table 321
Read Memory	R <address> <number of bytes>	Table 322
Prepare sector(s) for write operation	P <start sector number> <end sector number>	Table 323
Copy RAM to flash	C <Flash address> <RAM address> <number of bytes>	Table 324
Go	G <address> <Mode>	Table 325
Erase sector(s)	E <start sector number> <end sector number>	Table 326
Blank check sector(s)	I <start sector number> <end sector number>	Table 327
Read Part ID	J	Table 328
Read Boot code version	K	Table 330
Compare	M <address1> <address2> <number of bytes>	Table 331
ReadUID	N	Table 332

21.13.1 Unlock <Unlock code>

Table 318. ISP Unlock command

Command	U
Input	Unlock code: 23130_{10}
Return Code	CMD_SUCCESS | INVALID_CODE | PARAM_ERROR
Description	This command is used to unlock Flash Write, Erase, and Go commands.
Example	"U 23130<CR><LF>" unlocks the Flash Write/Erase & Go commands.

21.13.2 Set Baud Rate <Baud Rate> <stop bit>

Table 319. 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.

21.13.3 Echo <setting>

Table 320. 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.

21.13.4 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 than 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 321. 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.

21.13.5 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 than 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 322. 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. 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.

21.13.6 Prepare sector(s) for write operation <start sector number> <end sector number>

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

Table 323. 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.

21.13.7 Copy RAM to flash <Flash address> <RAM address> <no of bytes>

When writing to the flash, the following limitations apply:

1. The smallest amount of data that can be written to flash by the copy RAM to flash command is 256 byte (equal to one page).
2. One page consists of 16 flash words (lines), and the smallest amount that can be modified per flash write is one flash word (one line). This limitation follows from the application of ECC to the flash write operation, see Section 21.6.
3. To avoid write disturbance (a mechanism intrinsic to flash memories), an erase should be performed after following 16 consecutive writes inside the same page. Note that the erase operation then erases the entire sector.

   Remark: Once a page has been written to 16 times, it is still possible to write to other pages within the same sector without performing a sector erase (assuming that those pages have been erased previously).

Table 324. ISP Copy command

Command	C
Input	**Flash Address(DST):** Destination flash address where data bytes are to be written. The destination address should be a 256 byte boundary. **RAM Address(SRC):** Source RAM address from where data bytes are to be read. **Number of Bytes:** Number of bytes to be written. Should be 256 | 512 | 1024 | 4096.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not on word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR WRITE_OPERATION | BUSY | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to program the flash memory. The "Prepare Sector(s) for Write Operation" command should precede this command. The affected sectors are automatically protected again once the copy command is successfully executed. The boot block cannot be written by this command. This command is blocked when code read protection is enabled. Also see Section 21.6 for the number of bytes that can be written.
Example	"C 0 268467504 512<CR><LF>" copies 512 bytes from the RAM address 0x1000 0800 to the flash address 0.

21.13.8 Go <address> <mode>

Table 325. ISP Go command

Command	G
Input	**Address:** Flash or RAM address from which the code execution is to be started. This address should be on a word boundary. **Mode:** T (Execute program in Thumb Mode)
Return Code	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to execute a program residing in RAM or flash memory. It may not be possible to return to the ISP command handler once this command is successfully executed. This command is blocked when code read protection is enabled.
Example	"G 0 T<CR><LF>" branches to address 0x0000 0000.

21.13.9 Erase sector(s) <start sector number> <end sector number>

Table 326. 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.

21.13.10 Blank check sector(s) <sector number> <end sector number>

Table 327. ISP Blank check sector command

Command	I
Input	**Start Sector Number:** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | SECTOR_NOT_BLANK (followed by <Offset of the first non blank word location> <Contents of non blank word location>) | INVALID_SECTOR | PARAM_ERROR
Description	This command is used to blank check one or more sectors of on-chip flash memory. **Blank check on sector 0 always fails as first 64 bytes are re-mapped to flash boot block.**
Example	"I 2 3<CR><LF>" blank checks the flash sectors 2 and 3.

21.13.11 Read Part Identification number

Table 328. ISP Read Part Identification command

Command	J
Input	None.
Return Code	CMD_SUCCESS followed by device identification number in ASCII (see Table 329 “LPC13xx device identification numbers”).
Description	This command is used to read the device identification number.

Table 329. LPC13xx device identification numbers

Device	ASCII coding	Hex coding
LPC1311FHN33	742543403	0x2C42 502B
LPC1313FHN33	742395947	0x2C40 102B
LPC1313FBD48	742395947	0x2C40 102B
LPC1342FHN33	1023492139	0x3D01 402B
LPC1342FBD48	1023492139	0x3D01 402B
LPC1343FHN33	1023410219	0x3D00 002B
LPC1343FBD48	1023410219	0x3D00 002B
LPC1311FHN33/01	404131883	0x1816 902B
LPC1313FHN33/01	405803051	0x1830 102B
LPC1313FBD48/01	405803051	0x1830 102B

21.13.12 Read Boot code version number

Table 330. 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.

21.13.13 Compare <address1> <address2> <no of bytes>

Table 331. 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.

21.13.14 ReadUID

Table 332. ReadUID command

Command	N
Input	None
Return Code	CMD_SUCCESS followed by four 32-bit words of a unique serial number in ASCII format. The word sent at the lowest address is sent first.
Description	This command is used to read the unique ID.

21.13.15 ISP Return Codes

Table 333. 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.

21.14 IAP commands

For in application programming the IAP routine should be called with a word pointer in register r0 pointing to memory (RAM) containing command code and parameters. Result of the IAP command is returned in the result table pointed to by register r1. The user can reuse the command table for result by passing the same pointer in registers r0 and r1. The parameter table should be big enough to hold all the results in case the number of results are more than number of parameters. Parameter passing is illustrated in the Figure 64. 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 334. IAP Command Summary

IAP Command	Command Code	Described in
Prepare sector(s) for write operation	50 (decimal)	Table 335
Copy RAM to flash	51 (decimal)	Table 336
Erase sector(s)	52 (decimal)	Table 337
Blank check sector(s)	53 (decimal)	Table 338
Read Part ID	54 (decimal)	Table 339
Read Boot code version	55 (decimal)	Table 340
Compare	56 (decimal)	Table 341
Reinvoke ISP	57 (decimal)	Table 342
Read UID	58 (decimal)	Table 343

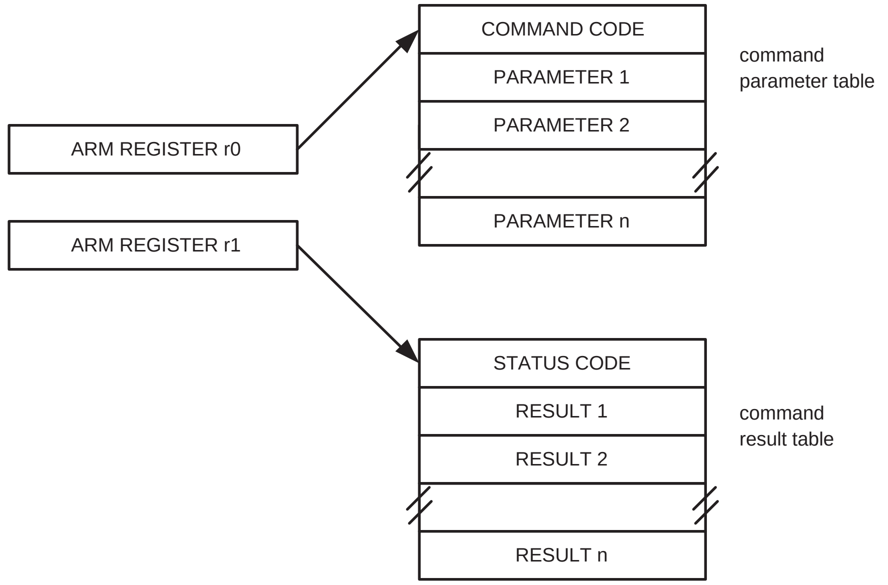


Fig 64. IAP parameter passing

21.14.1 Prepare sector(s) for write operation

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

Table 335. 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.

21.14.2 Copy RAM to flash

See Section 21.13.7 for limitations on the write-to-flash process.

Table 336. IAP Copy RAM to flash command

Command	Copy RAM to flash
Input	**Command code: 51 (decimal)** **Param0(DST):** Destination flash address where data bytes are to be written. This address should be a 256 byte boundary. **Param1(SRC):** Source RAM address from which data bytes are to be read. This address should be a word boundary. **Param2:** Number of bytes to be written. Should be 256 | 512 | 1024 | 4096. **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 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | BUSY
Result	None
Description	This command is used to program the flash memory. The affected sectors should be prepared first by calling "Prepare Sector for Write Operation" command. The affected sectors are automatically protected again once the copy command is successfully executed. The boot sector can not be written by this command. Also see Section 21.6 for the number of bytes that can be written.

21.14.3 Erase Sector(s)

Table 337. 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.

21.14.4 Blank check sector(s)

Table 338. 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.

21.14.5 Read Part Identification number

Table 339. 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 device identification number (see Table 329).

21.14.6 Read Boot code version number

Table 340. 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.

21.14.7 Compare <address1> <address2> <no of bytes>

Table 341. 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.**

21.14.8 Reinvoke ISP

Table 342. Reinvoke ISP

Command	Compare
Input	**Command code: 57 (decimal)**
Return Code	None
Result	**None.**
Description	This command is used to invoke the bootloader in ISP mode. It maps boot vectors, sets PCLK = CCLK, configures UART pins RXD and TXD, resets counter/timer CT32B1 and resets the U0FDR (see Table 209). This command may be used when a valid user program is present in the internal flash memory and the PIO0_1 pin is not accessible to force the ISP mode.

21.14.9 ReadUID

Table 343. IAP ReadUID command

Command	Compare
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.

21.14.10 IAP Status Codes

Table 344. IAP Status Codes Summary

Status Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on a word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable.
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data is not same.
11	BUSY	flash programming hardware interface is busy.

21.15 Debug notes

21.15.1 Comparing flash images

Depending on the debugger used and the IDE debug settings, the memory that is visible when the debugger connects might be the boot ROM, the internal SRAM, or the flash. To help determine which memory is present in the current debug environment, check the value contained at flash address 0x0000 0004. This address contains the entry point to the code in the ARM Cortex-M3 vector table, which is the bottom of the boot ROM, the internal SRAM, or the flash memory respectively.

Table 345. Memory mapping in debug mode

Memory mapping mode	Memory start address visible at 0x0000 0004
Bootloader mode	0x1FFF 0000
User flash mode	0x0000 0000
User SRAM mode	0x1000 0000

21.15.2 Serial Wire Debug (SWD) flash programming interface

Debug tools can write parts of the flash image to RAM and then execute the IAP call "Copy RAM to flash" repeatedly with proper offset.

21.16 Register description

Table 346. Register overview: FMC (base address 0x4003 C000)

Name	Access	Address offset	Description	Reset value	Reference
FLASHCFG	R/W	0x010	Flash configuration register	-	Table 347
FMSSTART	R/W	0x020	Signature start address register	0	Table 348
FMSSTOP	R/W	0x024	Signature stop-address register	0	Table 349
FMSW0	R	0x02C	Word 0 [31:0]	-	Table 350
FMSW1	R	0x030	Word 1 [63:32]	-	Table 351
FMSW2	R	0x034	Word 2 [95:64]	-	Table 352
FMSW3	R	0x038	Word 3 [127:96]	-	Table 353
FMSTAT	R	0xFE0	Signature generation status register	0	Section 21.16.4
FMSTATCLR	W	0xFE8	Signature generation status clear register	-	Section 21.16.5

21.16.1 Flash configuration register

Depending on the system clock frequency, access to the flash memory can be configured with various access times by writing to the FLASHCFG register at address 0x4003 C010.

Remark: Improper setting of this register may result in incorrect operation of the LPC13xx flash memory.

Table 347. Flash configuration register (FLASHCFG, address 0x4003 C010) bit description

Bit	Symbol	Valu e	Description	Reset value
1:0	FLASHTIM		Flash memory access time. FLASHTIM +1 is equal to the number of system clocks used for flash access.	10
		0x0	1 system clock flash access time (for system clock frequencies of up to 20 MHz).	
		0x1	2 system clocks flash access time (for system clock frequencies of up to 40 MHz).	
		0x2	3 system clocks flash access time (for system clock frequencies of up to 72 MHz).	
		0x3	Reserved.	
31:2	-	-	Reserved. **User software must not change the value of these bits. Bits 31:2 must be written back exactly as read**.	-

21.16.2 Signature generation address and control registers

These registers control automatic signature generation. A signature can be generated for any part of the flash memory contents. The address range to be used for generation is defined by writing the start address to the signature start address register (FMSSTART) and the stop address to the signature stop address register (FMSSTOP. The start and stop addresses must be aligned to 128-bit boundaries and can be derived by dividing the byte address by 16.

Signature generation is started by setting the SIG_START bit in the FMSSTOP register. Setting the SIG_START bit is typically combined with the signature stop address in a single write.

Table 348 and Table 349 show the bit assignments in the FMSSTART and FMSSTOP registers respectively.

Table 348. Flash Module Signature Start register (FMSSTART - 0x4003 C020) bit description

Bit	Symbol	Description	Reset value
16:0	START	Signature generation start address (corresponds to AHB byte address bits[20:4]).	0
31:17	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 349. Flash Module Signature Stop register (FMSSTOP - 0x4003 C024) bit description

Bit	Symbol	Value	Description	Reset value
16:0	STOP		BIST stop address divided by 16 (corresponds to AHB byte address [20:4]).	0
17	SIG_START		Start control bit for signature generation.	0
		0	Signature generation is stopped	
		1	Initiate signature generation	
31:18	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

21.16.3 Signature generation result registers

The signature generation result registers return the flash signature produced by the embedded signature generator. The 128-bit signature is reflected by the four registers FMSW0, FMSW1, FMSW2 and FMSW3.

The generated flash signature can be used to verify the flash memory contents. The generated signature can be compared with an expected signature and thus makes saves time and code space. The method for generating the signature is described in Section 21.17.1.

Table 353 show bit assignment of the FMSW0 and FMSW1, FMSW2, FMSW3 registers respectively.

Table 350. FMSW0 register bit description (FMSW0, address: 0x4003 C02C)

Bit	Symbol	Description	Reset value
31:0	SW0_31_0	Word 0 of 128-bit signature (bits 31 to 0).	-

Table 351. FMSW1 register bit description (FMSW1, address: 0x4003 C030)

Bit	Symbol	Description	Reset value
31:0	SW1_63_32	Word 1 of 128-bit signature (bits 63 to 32).	-

Table 352. FMSW2 register bit description (FMSW2, address: 0x4003 C034)

Bit	Symbol	Description	Reset value
31:0	SW2_95_64	Word 2 of 128-bit signature (bits 95 to 64).	-

Table 353. FMSW3 register bit description (FMSW3, address: 0x4003 40C8)

Bit	Symbol	Description	Reset value
31:0	SW3_127_96	Word 3 of 128-bit signature (bits 127 to 96).	-

21.16.4 Flash Module Status register

The read-only FMSTAT register provides a means of determining when signature generation has completed. Completion of signature generation can be checked by polling the SIG_DONE bit in FMSTAT. SIG_DONE should be cleared via the FMSTATCLR register before starting a signature generation operation, otherwise the status might indicate completion of a previous operation.

Table 354. Flash module Status register (FMSTAT - 0x4003 CFE0) 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	SIG_DONE	When 1, a previously started signature generation has completed. See FMSTATCLR register description for clearing this flag.	0
31:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

21.16.5 Flash Module Status Clear register

The FMSTATCLR register is used to clear the signature generation completion flag.

Table 355. Flash Module Status Clear register (FMSTATCLR - 0x0x4003 CFE8) 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	SIG_DONE_CLR	Writing a 1 to this bits clears the signature generation completion flag (SIG_DONE) in the FMSTAT register.	0
31:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

21.17 Flash signature generation

The flash module contains a built-in signature generator. This generator can produce a 128-bit signature from a range of flash memory. A typical usage is to verify the flashed contents against a calculated signature (e.g. during programming).

The address range for generating a signature must be aligned on flash-word boundaries, i.e. 128-bit boundaries. Once started, signature generation completes independently. While signature generation is in progress, the flash memory cannot be accessed for other purposes, and an attempted read will cause a wait state to be asserted until signature generation is complete. Code outside of the flash (e.g. internal RAM) can be executed during signature generation. This can include interrupt services, if the interrupt vector table is re-mapped to memory other than the flash memory. The code that initiates signature generation should also be placed outside of the flash memory.

21.17.1 Algorithm and procedure for signature generation

Signature generation

A signature can be generated for any part of the flash contents. The address range to be used for signature generation is defined by writing the start address to the FMSSTART register, and the stop address to the FMSSTOP register.

The signature generation is started by writing a ‘1’ to FMSSTOP.MISR_START. Starting the signature generation is typically combined with defining the stop address, which is done in another field FMSSTOP.FMSSTOP of the same register.

The time that the signature generation takes is proportional to the address range for which the signature is generated. Reading of the flash memory for signature generation uses a self-timed read mechanism and does not depend on any configurable timing settings for the flash. A safe estimation for the duration of the signature generation is:

Duration = int((60 / tcy) + 3) x (FMSSTOP - FMSSTART + 1)

When signature generation is triggered via software, the duration is in AHB clock cycles, and tcy is the time in ns for one AHB clock. The SIG_DONE bit in FMSTAT can be polled by software to determine when signature generation is complete.

If signature generation is triggered via JTAG, the duration is in JTAG tck cycles, and tcy is the time in ns for one JTAG clock. Polling the SIG_DONE bit in FMSTAT is not possible in this case.

After signature generation, a 128-bit signature can be read from the FMSW0 to FMSW3 registers. The 128-bit signature reflects the corrected data read from the flash. The 128-bit signature reflects flash parity bits and check bit values.

Content verification

The signature as it is read from the FMSW0 to FMSW3 registers must be equal to the reference signature. The algorithms to derive the reference signature is given in Figure 65.

```
sign = 0
FOR address = FMSTART.FMSTART TO FMSTOP.FMSTOP
{
    FOR i = 0 TO 126
        nextSign[i] = f_Q[address][i] XOR sign[i+1]
        nextSign[127] = f_Q[address][127] XOR sign[0] XOR sign[2] XOR
            sign[27] XOR sign[29]
    sign = nextSign
}
signature128 = sign
```

Fig 65. Algorithm for generating a 128 bit signature

22.1 How to read this chapter

The debug functionality is identical for all LPC13xx parts.

22.2 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.
- Trace port provides CPU instruction trace capability. Output via a Serial Wire Viewer.
- Eight breakpoints. Six instruction breakpoints that can also be used to remap instruction addresses for code patches. Two data comparators that can be used to remap addresses for patches to literal values.
- Four data watchpoints that can also be used as trace triggers.
- Instrumentation Trace Macrocell allows additional software controlled trace.

22.3 Introduction

Debug and trace functions are integrated into the ARM Cortex-M3. Serial wire debug and trace functions are supported. The ARM Cortex-M3 is configured to support up to eight breakpoints and four watchpoints.

22.4 Description

Debugging with the LPC13xx uses the Serial Wire Debug mode.

Trace can be done using the Serial Wire Output. When the Serial Wire Output is used, less data can be traced, but it uses no application related pins. Note that the trace function available for the ARM Cortex-M3 is functionally very different than the trace that was available for previous ARM7 based devices.

22.5 Pin description

The tables below indicate the various pin functions related to debug and trace. Some of these functions share pins with other functions which therefore may not be used at the same time. Trace using the Serial Wire Output has limited bandwidth.

Table 356. 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 LPC13xx.
SWO	Output	**Serial Wire Output.** The SWO pin optionally provides data from the ITM and/or the ETM for an external debug tool to evaluate.

22.6 Debug notes

22.6.1 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-M3 integration, the LPC13xx cannot wake up in the usual manner from Deep-sleep mode. It is recommended not to use this mode during debug.

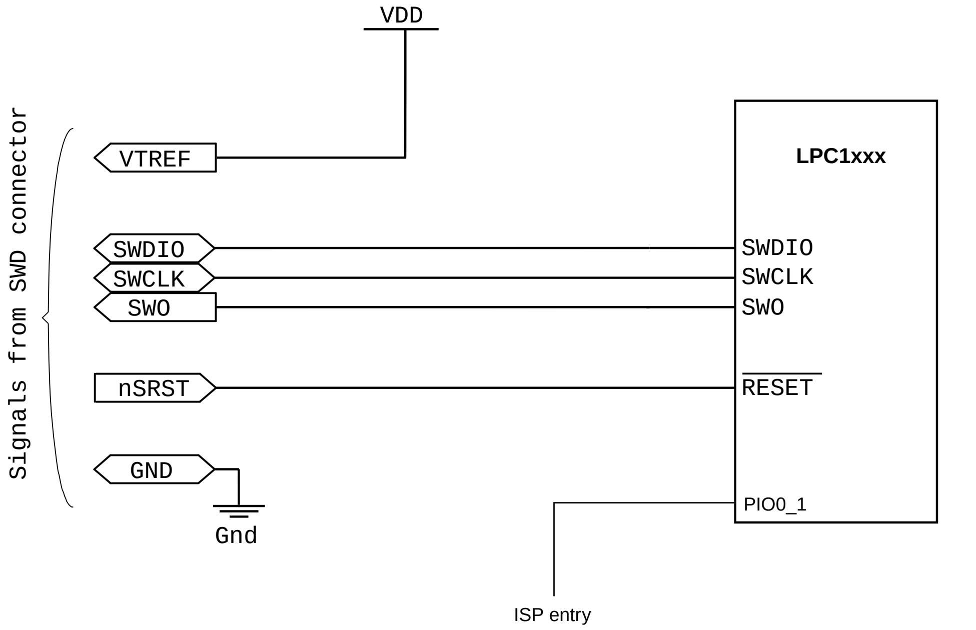
Another issue is that debug mode changes the way in which reduced power modes work internal to the ARM Cortex-M3 CPU, and this ripples through the entire system. These differences mean that power measurements should not be made while debugging, the results will be higher than during normal operation in an application.

During a debugging session, the System Tick Timer is automatically stopped whenever the CPU is stopped. Other peripherals are not affected.

Remark: Note that the debug mode is not supported in any of the reduced power modes.

22.6.2 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 66. Connecting the SWD pins to a standard SWD connector

UM10375

Chapter 23: LPC13xx Supplementary information

Rev. 5 — 21 June 2012 User manual

23.1 Abbreviations

Table 357. Abbreviations

Acronym	Description
A/D	Analog-to-Digital
AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
BOD	BrownOut Detection
DCC	Debug Communication Channel
DSP	Digital Signal Processing
EOP	End Of Packet
ETM	Embedded Trace Macrocell
GPIO	General Purpose Input/Output
I/O	Input/Output
MSC	Mass Storage Class
PHY	Physical Layer
PLL	Phase-Locked Loop
SE0	Single Ended Zero
SPI	Serial Peripheral Interface
SSI	Serial Synchronous Interface
SoF	Start-of-Frame
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

23.2 Legal information

23.2.1 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.

23.2.2 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.

23.2.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

I^{2}C-bus — logo is a trademark of NXP B.V.

23.3 Tables

23.4 Figures

23.5 Contents

Chapter 1: LPC13xx Introductory information

Chapter 2: LPC13xx Memory mapping

Chapter 3: LPC13xx System configuration

Chapter 4: LPC13xx Power Management Unit (PMU)

Chapter 5: LPC13xx Power profiles

Chapter 6: LPC13xx Interrupt controller

Chapter 7: LPC13xx I/O configuration

Chapter 8: LPC13xx Pin configuration

Chapter 9: LPC13xx General Purpose I/O (GPIO)

Chapter 10: LPC13xx USB device controller

Chapter 11: LPC13xx USB on-chip drivers

Chapter 12: LPC13xx UART

Chapter 13: LPC13xx I2C-bus controller

Chapter 14: LPC13xx SSP0/1

Chapter 15: LPC13xx 16-bit timer/counters (CT16B0/1)

Chapter 16: LPC13xx 32-bit timer/counters (CT32B0/1)

Chapter 17: LPC13xx System tick timer

Chapter 18: LPC13xx WatchDog Timer (WDT)

Chapter 19: LPC13xx Windowed WatchDog Timer (WWDT)

Chapter 20: LPC13xx Analog-to-Digital Converter (ADC)

Chapter 21: LPC13xx Flash memory programming firmware

Chapter 22: LPC13xx Serial Wire Debug (SWD)

Chapter 23: LPC13xx Supplementary information

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section ‘Legal 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: 21 June 2012
Document identifier: UM10375