UM10601

LPC800 User manual

Rev. 1.2 — 14 March 2013 User manual

Document information

Info	Content
Keywords	ARM Cortex M0+, LPC800, USART, I2C, LPC810M021FN8, LPC811M001FDH16, LPC812M101FDH16, LPC812M101FD20, LPC812M101FDH20
Abstract	LPC800 user manual

Revision history

Rev	Date	Description
1.2	20130314	LPC800 user manual
Modifications:		• Editorial updates. • Table 40 “PLL configuration examples” updated. • Register bit description of Table 92 “Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description” updated. • Chapter 5 “LPC800 Reduced power modes and Power Management Unit (PMU)” updated. • Section 5.3.1 “Low power modes in the ARM Cortex-M0+ core” added. • Removed dependency on system frequency for flash access times in Table 213 “Flash configuration register (FLASHCFG, address 0x4004 0010) bit description”. • Instructions on how to prevent floating internal pins added. See Section 6.3. • Figure 30 “I2C clocking” updated. • Description of the NMISRC register updated. See Section 4.6.26 “NMI source selection register”. • Section 16.3.1 “I2C transmit/receive in master mode” added. • Chapter 14 “LPC800 ARM Cortex SysTick Timer (SysTick)” added. • Address offset of the DEVICE_ID register corrected. See Table 38 “Device ID register (DEVICE_ID, address 0x4004 83F8) bit description”. • BOD reset level 0 changed to reserved in Table 28 “BOD control register (BODCTRL, address 0x4004 8150) bit description”.
1.1	20130124	LPC800 user manual
Modifications:		• Flash signature creation algorithm corrected. See Section 19.5.1 “Flash signature generation”. • System PLL output frequency restricted to < 100 MHz. • MTB register memory space changed to 1 kB in Figure 2 “LPC800 Memory mapping”. • Description of the External trace buffer command register updated. See Section 4.6.20 “External trace buffer command register”. • Flash interrupt removed in Table 3. • Chapter 27 summarizing the ARM Cortex-M0+ instruction set added. • ISP Read CRC checksum command added. See Section 21.5.1.15 “Read CRC checksum <address> <no of bytes>”. • Section 20.3.1 “Boot loader versions” added. • MRT implementation changed to 31-bit timer. See Chapter 11. Bit description of Table 140 “Idle channel register (IDLE_CH, address 0x4000 40F4) bit description” corrected. • Updates for clarification in Chapter 17 “LPC800 SPI0/1”. • Updates for clarification in Chapter 16 “LPC800 I2C-bus interface”. • Updates for clarification in Chapter 15 “LPC800 USART0/1/2”. • Updates for clarification in Chapter 8 “LPC800 Pin interrupts/pattern match engine”. • Updates for clarification in Section 9.4 (switch matrix-to-pin functional diagram). • Updates for clarification in Chapter 5 “LPC800 Reduced power modes and Power Management Unit (PMU)”. • Section 3.3.2 “Non-Maskable Interrupt (NMI)” and Section 3.3.3 “Vector table offset” added. • Bit fields corrected in Section 10.6. • USART baudrate clock output removed from USART features.

Revision history *...continued*

Rev	Date	Description
1	20121109	Preliminary LPC800 user manual

Contact information

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

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

1.1 Introduction

The LPC800 are an ARM Cortex-M0+ based, low-cost 32-bit MCU family operating at CPU frequencies of up to 30 MHz. The LPC800 support up to 16 kB of flash memory and 4 kB of SRAM.

The peripheral complement of the LPC800 includes a CRC engine, one I^2C-bus interface, up to three USARTs, up to two SPI interfaces, one multi-rate timer, self wake-up timer, and state-configurable timer, one comparator, function-configurable I/O ports through a switch matrix, an input pattern match engine, and up to 18 general-purpose I/O pins.

1.2 Features

- System:
  - ARM Cortex-M0+ processor, running at frequencies of up to 30 MHz with single-cycle multiplier and fast single-cycle I/O port.
  - ARM Cortex-M0+ built-in Nested Vectored Interrupt Controller (NVIC).
  - System tick timer.
  - Serial Wire Debug (SWD) and JTAG boundary scan modes supported.
  - Micro Trace Buffer (MTB) supported.
- Memory:
  - Up to 16 kB on-chip flash programming memory with 64 Byte page write and erase.
  - 4 kB SRAM.
- ROM API support:
  - Boot loader.
  - USART drivers.
  - I2C drivers.
  - Power profiles.
  - Flash In-Application Programming (IAP) and In-System Programming (ISP).
- Digital peripherals:
  - High-speed GPIO interface connected to the ARM Cortex-M0+ IO bus with up to 18 General-Purpose I/O (GPIO) pins with configurable pull-up/pull-down resistors, programmable open-drain mode, input inverter, and glitch filter.
  - High-current source output driver (20 mA) on four pins.
  - High-current sink driver (20 mA) on two true open-drain pins.
  - GPIO interrupt generation capability with boolean pattern-matching feature on eight GPIO inputs.
  - Switch matrix for flexible configuration of each I/O pin function.

  - State Configurable Timer (SCT) with input and output functions (including capture and match) assigned to pins through the switch matrix.
  - Multiple-channel multi-rate timer (MRT) for repetitive interrupt generation at up to four programmable, fixed rates.
  - Self Wake-up Timer (WKT) clocked from either the IRC or a low-power, low-frequency internal oscillator.
  - CRC engine.
  - Windowed Watchdog timer (WWDT).
- Analog peripherals:
  - Comparator with external voltage reference with pin functions assigned or enabled through the switch matrix.
- Serial interfaces:
  - Three USART interfaces with pin functions assigned through the switch matrix.
  - Two SPI controllers with pin functions assigned through the switch matrix.
  - One I^2C-bus interface with pin functions assigned through the switch matrix.
- Clock generation:
  - 12 MHz internal RC oscillator trimmed to 1 % accuracy that can optionally be used as a system clock.
  - Crystal oscillator with an operating range of 1 MHz to 25 MHz.
  - Programmable watchdog oscillator with a frequency range of 9.4 kHz to 2.3 MHz.
  - 10 kHz low-power oscillator for the WKT.
  - PLL allows CPU operation up to the maximum CPU rate without the need for a high-frequency crystal. May be run from the system oscillator, the external clock input CLKIN, or the internal RC oscillator.
  - Clock output function with divider that can reflect the crystal oscillator, the main clock, the IRC, or the watchdog oscillator.
- Power control:
  - Integrated PMU (Power Management Unit) to minimize power consumption.
  - Reduced power modes: Sleep mode, Deep-sleep mode, Power-down mode, and Deep power-down mode.
  - Wake-up from Deep-sleep and Power-down modes on activity on USART, SPI, and I2C peripherals.
  - Timer-controlled self wake-up from Deep power-down mode.
  - Power-On Reset (POR).
  - Brownout detect.
- Unique device serial number for identification.
- Single power supply.
- Available as SO20 package, TSSOP20 package, TSSOP16, and DIP8 package.

1.3 Ordering information

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
LPC810M021FN8	DIP8	plastic dual in-line package; 8 leads (300 mil)	SOT097-2
LPC811M001FDH16	TSSOP16	plastic thin shrink small outline package; 16 leads; body width 4.4 mm	SOT403-1
LPC812M101FDH16	TSSOP16	plastic thin shrink small outline package; 16 leads; body width 4.4 mm	SOT403-1
LPC812M101FD20	SO20	plastic small outline package; 20 leads; body width 7.5 mm	SOT163-1
LPC812M101FDH20	TSSOP20	plastic thin shrink small outline package; 20 leads; body width 4.4 mm	SOT360-1

Table 2. Ordering options

Type number	Flash/kB	SRAM/kB	USART	I^2C	SPI	Comparator	GPIO	Package
LPC810M021FN8	4	1	2	1	1	1	6	DIP8
LPC811M001FDH16	8	2	2	1	1	1	14	TSSOP16
LPC812M101FDH16	16	4	3	1	2	1	14	TSSOP16
LPC812M101FD20	16	4	2	1	1	1	18	SO20
LPC812M101FDH20	16	4	3	1	2	1	18	TSSOP20

1.4 Block diagram

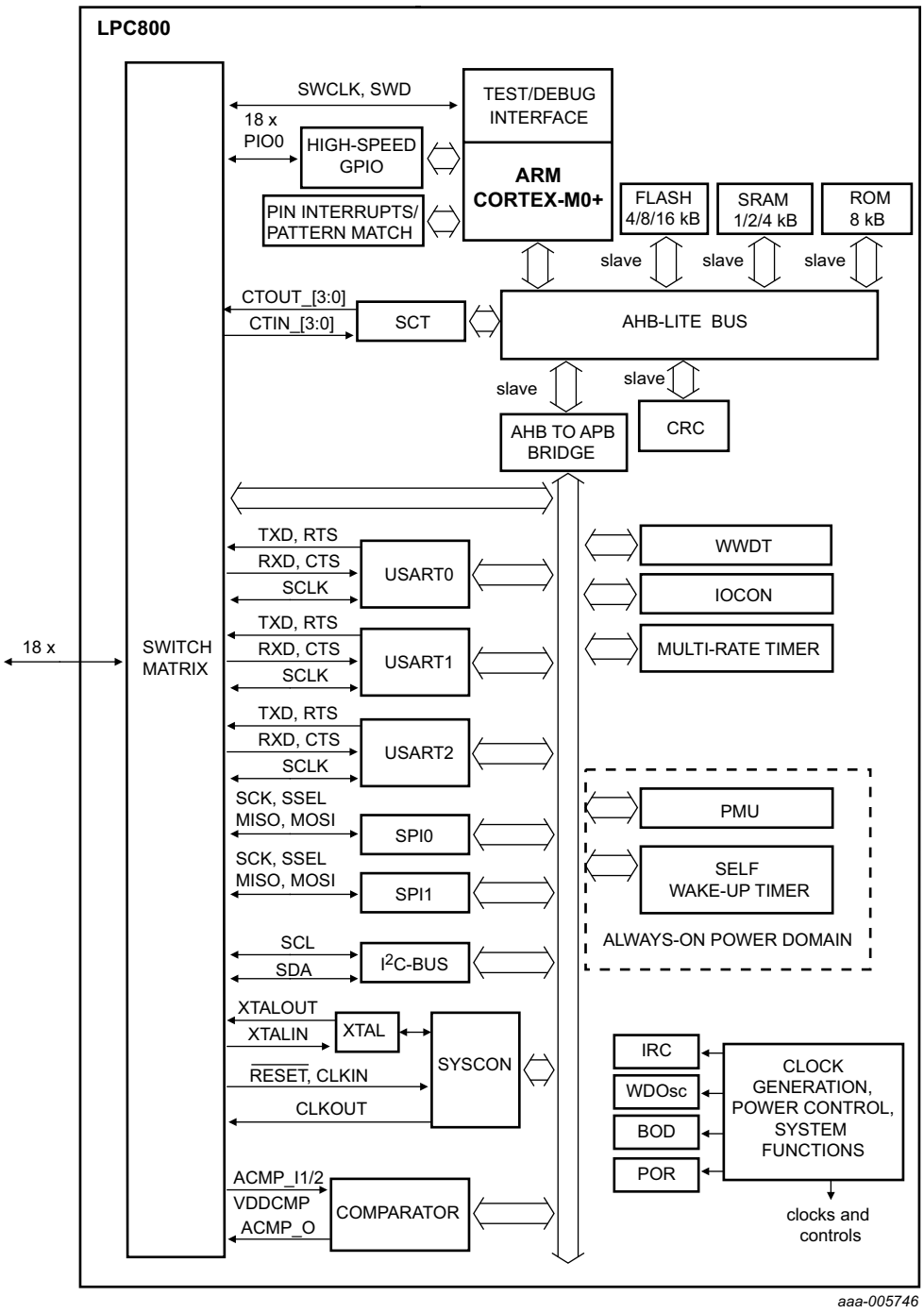


Fig 1. LPC800 block diagram

1.5 General description

1.5.1 ARM Cortex-M0+ core configuration

The ARM Cortex-M0+ core runs at an operating frequency of up to 30 MHz. Integrated in the core are the NVIC and Serial Wire Debug with four breakpoints and two watch points. The ARM Cortex-M0+ core supports a single-cycle I/O enabled port (IOP) for fast GPIO access at address 0xA000 0000.

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

2.1 How to read this chapter

The memory mapping is identical for all LPC800 parts. Different LPC800 parts support different flash memory sizes.

2.2 General description

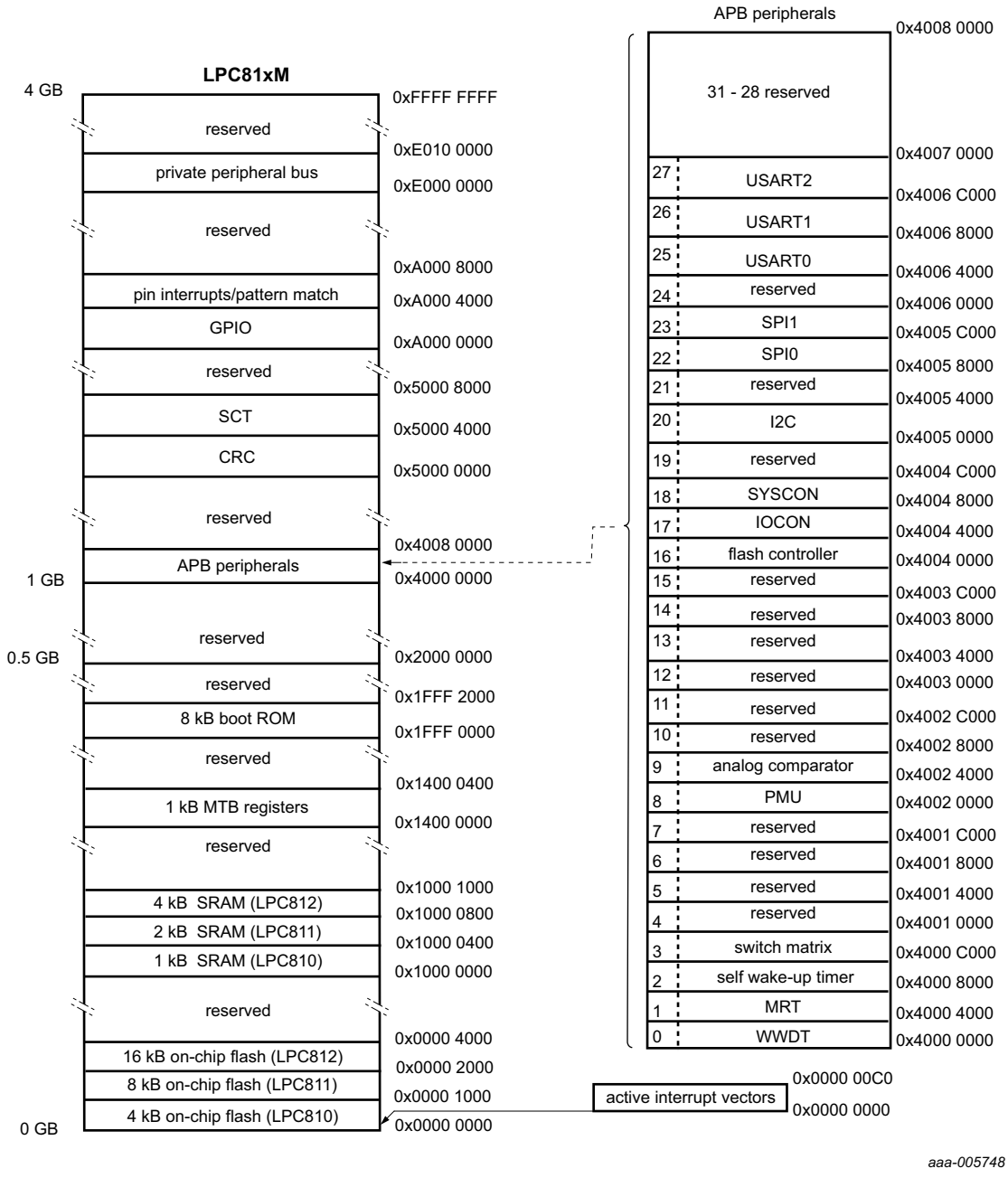
The LPC800 incorporates several distinct memory regions. Figure 2 shows the overall map of the entire address space from the user program viewpoint following reset.

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

The registers incorporated into the ARM Cortex-M0+ core, such as NVIC, SysTick, and sleep mode control, are located on the private peripheral bus.

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

2.2.1 Memory mapping



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

Fig 2. LPC800 Memory mapping

2.2.2 Micro Trace Buffer (MTB)

The LPC800 supports the ARM Cortex-M0+ Micro Trace Buffer. See Section 26.5.4.

3.1 How to read this chapter

The NVIC is identical on all LPC800 parts.

The SPI1 and USART2 interrupts are implemented on parts LPC812M101FDH20 and LPC812M101FDH16 only.

3.2 Features

- Nested Vectored Interrupt Controller that is an integral part of the ARM Cortex-M0+.
- Tightly coupled interrupt controller provides low interrupt latency.
- Controls system exceptions and peripheral interrupts.
- The NVIC supports 32 vectored interrupts.
- Four programmable interrupt priority levels with hardware priority level masking.
- Software interrupt generation using the ARM exceptions SVCall and PendSV (see Ref. 1).
- Support for NMI.
- ARM Cortex M0+ Vector table offset register VTOR implemented.

3.3 General description

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

3.3.1 Interrupt sources

Table 3 lists the interrupt sources for each peripheral function. Each peripheral device may have one or more interrupt lines to the Vectored Interrupt Controller. Each line may represent more than one interrupt source. Interrupts with the same priority level are serviced in the order of their interrupt number.

See Ref. 1 for a detailed description of the NVIC and the NVIC register description.

Table 3. Connection of interrupt sources to the NVIC

Interrupt number	Name	Description	Flags
0	SPI0_IRQ	SPI0 interrupt	See Table 194 “SPI Interrupt Enable read and Set register (INTENSET, addresses 0x4005 800C (SPI0) , 0x4005 C00C (SPI1)) bit description”.
1	SPI1_IRQ	SPI1 interrupt	Same as SPI0_IRQ
2	-	Reserved	-

Table 3. Connection of interrupt sources to the NVIC

Interrupt number	Name	Description	Flags
3	UART0_IRQ	USART0 interrupt	See Table 163 “USART Interrupt Enable read and set register (INTENSET, address 0x4006 400C (USART0), 0x4006 800C (USART1), 0x4006 C00C (USART2)) bit description”
4	UART1_IRQ	USART1 interrupt	Same as UART0_IRQ
5	UART2_IRQ	USART2 interrupt	Same as UART0_IRQ
6	-	Reserved	-
7	-	Reserved	-
8	I2C0_IRQ	I2C0 interrupt	See Table 177 “Interrupt Enable Clear register (INTENCLR, address 0x4005 000C) bit description”.
9	SCT_IRQ	State configurable timer interrupt	EVFLAG SCT event
10	MRT_IRQ	Multi-rate timer interrupt	Global MRT interrupt. GFLAG0 GFLAG1 GFLAG2 GFLAG3
11	CMP_IRQ	Analog comparator interrupt	COMPEDGE - rising, falling, or both edges can set the bit
12	WDT_IRQ	Windowed watchdog timer interrupt	WARNINT - watchdog warning interrupt
13	BOD_IRQ	BOD interrupts	BODINTVAL - BOD interrupt level
14	-	-	Reserved
15	WKT_IRQ	Self wake-up timer interrupt	ALARMFLAG
23:16	-	Reserved	-
24	PININT0_IRQ	Pin interrupt 0 or pattern match engine slice 0 interrupt	PSTAT - pin interrupt status
25	PININT1_IRQ	Pin interrupt 1 or pattern match engine slice 1 interrupt	PSTAT - pin interrupt status
26	PININT2_IRQ	Pin interrupt 2 or pattern match engine slice 2 interrupt	PSTAT - pin interrupt status
27	PININT3_IRQ	Pin interrupt 3 or pattern match engine slice 3 interrupt	PSTAT - pin interrupt status
28	PININT4_IRQ	Pin interrupt 4 or pattern match engine slice 4 interrupt	PSTAT - pin interrupt status

Table 3. Connection of interrupt sources to the NVIC

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

3.3.2 Non-Maskable Interrupt (NMI)

The LPC800 supports the NMI, which can be triggered by an peripheral interrupt or triggered by software. The NMI has the highest priority exception other than the reset.

You can set up any peripheral interrupt listed in Table 3 as NMI using the NMISRC register in the SYSCON block (Table 31). To avoid using the same peripheral interrupt as NMI exception and normal interrupt, disable the interrupt in the NVIC when you configure it as NMI.

3.3.3 Vector table offset

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

4.1 How to read this chapter

The system configuration block is identical for all LPC800 parts. USART2 and SPI1 are only available on parts LPC812M101FDH20 and LPC812M101FDH16 and the corresponding clocks, reset, and wake-up control bits are reserved for all other parts.

4.2 Features

- Clock control
  - Configure the system PLL.
  - Configure system oscillator and watchdog oscillator.
  - Enable clocks to individual peripherals and memories.
  - Configure clock output.
  - Configure clock dividers, digital filter clock, and USART baud rate clock.
- Monitor and release reset to individual peripherals.
- Select pins for external pin interrupts and pattern match engine.
- Configuration of reduced power modes.
- Wake-up control.
- BOD configuration.
- MTB trace start and stop.
- Interrupt latency control.
- Select a source for the NMI.
- Calibrate system tick timer.

4.3 Basic configuration

Configure the SYSCON block as follows:

- The SYSCON uses the CLKIN, CLKOUT, $\overline{\text{RESET}}$, and XTALIN/OUT pins. Configure the pin functions through the switch matrix. See Section 4.4.
- No clock configuration is needed. The clock to the SYSCON block is always enabled. By default, the SYSCON block is clocked by the IRC.

4.3.1 Set up the PLL

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

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

   Section 4.6.32 “Power configuration register”

2. Select the PLL input in the SYSPLLCLKSEL register. You have the following input options:
   - IRC: 12 MHz internal oscillator.
   - System oscillator: External crystal oscillator using the XTALIN/XTALOUT pins.
   - External clock input CLKIN. Select this pin through the switch matrix.

   Section 4.6.8 “System PLL clock source select register”
3. Update the PLL clock source in the SYSPLLCLKUEN register.

   Section 4.6.9 “System PLL clock source update register”
4. Configure the PLL M and N dividers.

   Section 4.6.3 “System PLL control register”
5. Wait for the PLL to lock by monitoring the PLL lock status.

   Section 4.6.4 “System PLL status register”

4.3.2 Configure the main clock and system clock

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

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

1. Select the main clock . You have the following options:
   - IRC: 12 MHz internal oscillator (default).
   - PLL output: You must configure the PLL to use the PLL output.

   Section 4.6.10 “Main clock source select register”
2. Update the main clock source.

   Section 4.6.11 “Main clock source update enable register”
3. Select the divider value for the system clock. A divider value of 0 disables the system clock.

   Section 4.6.12 “System clock divider register”
4. Select the memories and peripherals that are operating in your application and therefore must have an active clock. The core is always clocked.

   Section 4.6.13 “System clock control register”

4.3.3 Set up the system oscillator using XTALIN and XTALOUT

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

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

Related registers:

Table 63 “PIO0_8 register (PIO0_8, address 0x4004 4038) bit description”

Table 62 “PIO0_9 register (PIO0_9, address 0x4004 4034) bit description”

Table 106 “Pin enable register 0 (PINENABLE0, address 0x4000 C1C0) bit description”

Table 10 “System oscillator control register (SYSOSCCTRL, address 0x4004 8020) bit description”

4.4 Pin description

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

See Section 9.3.1 “Connect an internal signal to a package pin” to assign the CLKOUT function to a pin on the LPC800 package.

See Section 9.3.2 to enable the clock input, the oscillator pins, and the external reset input.

Table 4. SYSCON pin description

Function	Direction	Pin	Description	SWM register	Reference
CLKOUT	O	any	CLKOUT clock output.	PINASSIGN8	Table 105
CLKIN	I	PIO0_1/ACMP_I2/CLKIN	External clock input to the system PLL. Disable the ACMP_I2 function in the PINENABLE register.	PINENABLE0	Table 106
XTALIN	I	PIO0_8/XTALIN	Input to the system oscillator.	PINENABLE0	Table 106
XTALOUT	O	PIO0_9/XTALOUT	Output from the system oscillator.	PINENABLE0	Table 106
$\overline{\text{RESET}}$	I	$\overline{\text{RESET}}$/PIO0_5	External reset input	PINENABLE0	Table 106

4.5 General description

4.5.1 Clock generation

The system control block generates all clocks for the chip. Only the low-power oscillator used for wake-up timing is controlled by the PMU. Except for the USART clock and the clock to configure the glitch filters of the digital I/O pins, the clocks to the core and peripherals run at the same frequency. The maximum system clock frequency is 30 MHz. See Figure 3.

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

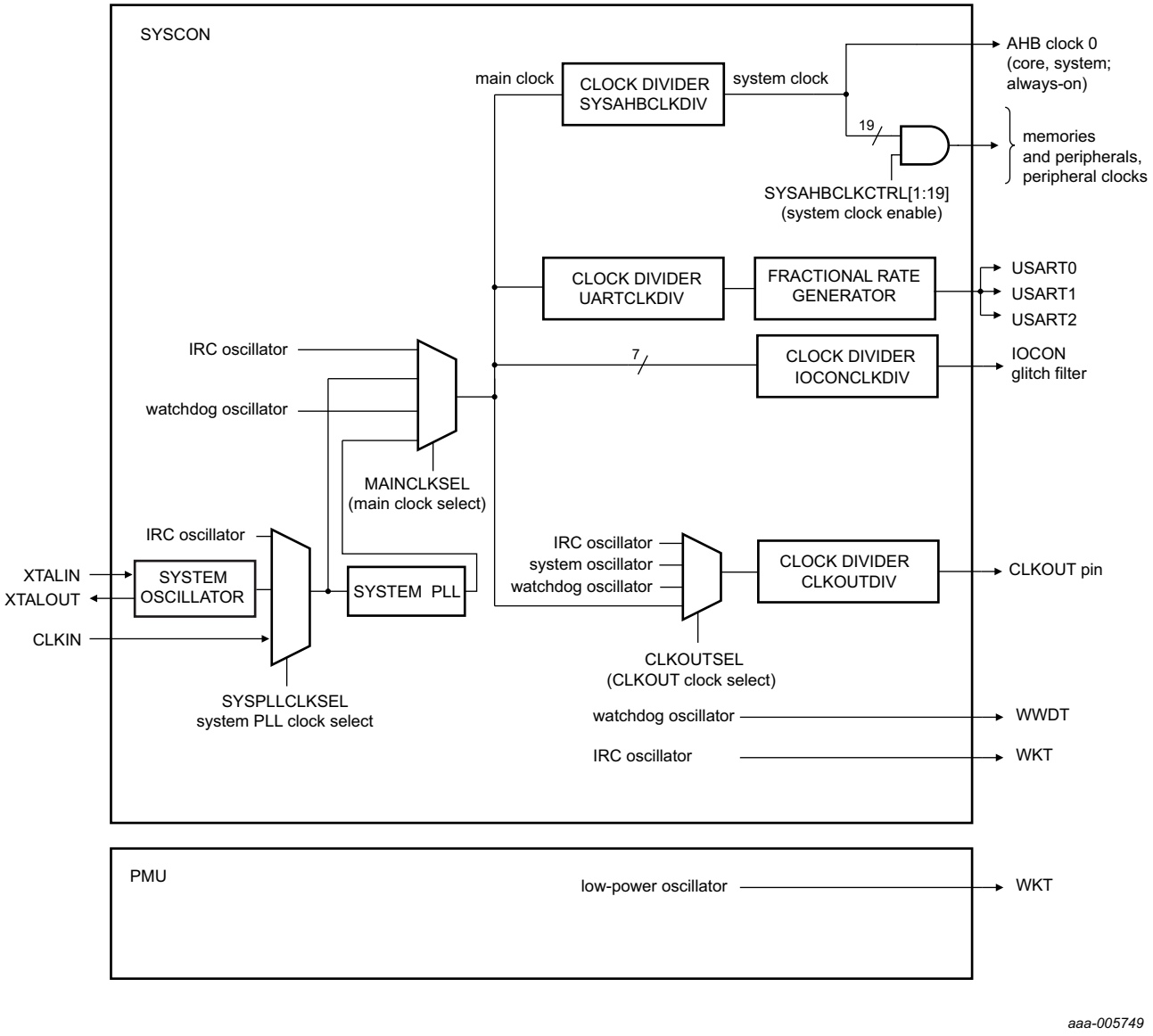


Fig 3. LPC800 clock generation

4.5.2 Power control of analog components

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

Section 4.6.30 “Deep-sleep mode configuration register”

Section 4.6.3 “System PLL control register”

Section 4.6.6 “Watchdog oscillator control register”

Section 4.6.5 “System oscillator control register”

4.5.3 Configuration of reduced power-modes

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

Section 4.6.32 “Power configuration register”

Section 4.6.29 “Start logic 1 interrupt wake-up enable register”

4.5.4 Reset and interrupt control

The peripheral reset control register in the system control register allows to assert and release individual peripheral resets. See Table 7.

Up to eight external pin interrupts can be assigned to any digital pin in the system control block (see Section 4.6.27 “Pin interrupt select registers”).

4.6 Register description

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

Reset values describe the content of the registers after the boot loader has executed.

All address offsets not shown in Table 5 are reserved and should not be written to.

Table 5. Register overview: System configuration (base address 0x4004 8000)

Name	Access	Offset	Description	Reset value	Reference
SYSMEMREMAP	R/W	0x000	System memory remap	0x2	Table 6
PRESETCTRL	R/W	0x004	Peripheral reset control	0x0000 1FFF	Table 7
SYSPLLCTRL	R/W	0x008	System PLL control	0	Table 8
SYSPLLSTAT	R	0x00C	System PLL status	0	Table 9
-	-	0x010	Reserved	-	-
-	-	0x014	Reserved	-	-
SYSOSCCTRL	R/W	0x020	System oscillator control	0x000	Table 10
WDTOSCCTRL	R/W	0x024	Watchdog oscillator control	0x0A0	Table 11
-	-	0x028	Reserved	-	-
-	-	0x02C	Reserved	-	-
SYSRSTSTAT	R/W	0x030	System reset status register	0	Table 12
SYSPLLCLKSEL	R/W	0x040	System PLL clock source select	0	Table 13
SYSPLLCLKUEN	R/W	0x044	System PLL clock source update enable	0	Table 14
MAINCLKSEL	R/W	0x070	Main clock source select	0	Table 15
MAINCLKUEN	R/W	0x074	Main clock source update enable	0	Table 16
SYSAHBCLKDIV	R/W	0x078	System clock divider	1	Table 17
SYSAHBCLKCTRL	R/W	0x080	System clock control	0x1F	Table 18
UARTCLKDIV	R/W	0x094	USART clock divider	0	Table 19
-	-	0x098	Reserved	-	-

Table 5. Register overview: System configuration (base address 0x4004 8000) *…continued*

Name	Access	Offset	Description	Reset value	Reference
-	-	0x09C	Reserved	-	-
-	-	0x0A0 - 0x0BC	Reserved	-	-
-	-	0x0CC	Reserved	-	-
CLKOUTSEL	R/W	0x0E0	CLKOUT clock source select	0	Table 20
CLKOUTUEN	R/W	0x0E4	CLKOUT clock source update enable	0	Table 21
CLKOUTDIV	R/W	0x0E8	CLKOUT clock divider	0	Table 22
UARTFRGDIV	R/W	0x0F0	USART fractional generator divider value	0	Table 23
UARTFRGMULT	R/W	0x0F4	USART fractional generator multiplier value	0	Table 24
EXTTRACECMD	R/W	0x0FC	External trace buffer command register	0	Table 25
PIOPORCAP0	R	0x100	POR captured PIO status 0	user dependent	Table 26
-	-	0x104	Reserved	-	-
IOCONCLKDIV6	R/W	0x134	Peripheral clock 6 to the IOCON block for programmable glitch filter	0x0000 0000	Table 27
IOCONCLKDIV5	R/W	0x138	Peripheral clock 5 to the IOCON block for programmable glitch filter	0x0000 0000	Table 27
IOCONCLKDIV4	R/W	0x13C	Peripheral clock 4 to the IOCON block for programmable glitch filter	0x0000 0000	Table 27
IOCONCLKDIV3	R/W	0x140	Peripheral clock 3 to the IOCON block for programmable glitch filter	0x0000 0000	Table 27
IOCONCLKDIV2	R/W	0x144	Peripheral clock 2 to the IOCON block for programmable glitch filter	0x0000 0000	Table 27
IOCONCLKDIV1	R/W	0x148	Peripheral clock 1 to the IOCON block for programmable glitch filter	0x0000 0000	Table 27
IOCONCLKDIV0	R/W	0x14C	Peripheral clock 0 to the IOCON block for programmable glitch filter	0x0000 0000	Table 27
BODCTRL	R/W	0x150	Brown-Out Detect	0	Table 28
SYSTCKCAL	R/W	0x154	System tick counter calibration	0x0	Table 29
-	R/W	0x168	Reserved	-	-
IRQLATENCY	R/W	0x170	IQR delay. Allows trade-off between interrupt latency and determinism.	0x0000 0010	Table 30
NMISRC	R/W	0x174	NMI Source Control	0	Table 31
PINTSEL0	R/W	0x178	GPIO Pin Interrupt Select register 0	0	Table 32
PINTSEL1	R/W	0x17C	GPIO Pin Interrupt Select register 1	0	Table 32
PINTSEL2	R/W	0x180	GPIO Pin Interrupt Select register 2	0	Table 32
PINTSEL3	R/W	0x184	GPIO Pin Interrupt Select register 3	0	Table 32
PINTSEL4	R/W	0x188	GPIO Pin Interrupt Select register 4	0	Table 32
PINTSEL5	R/W	0x18C	GPIO Pin Interrupt Select register 5	0	Table 32
PINTSEL6	R/W	0x190	GPIO Pin Interrupt Select register 6	0	Table 32
PINTSEL7	R/W	0x194	GPIO Pin Interrupt Select register 7	0	Table 32
STARTERP0	R/W	0x204	Start logic 0 pin wake-up enable register	0	Table 33
STARTERP1	R/W	0x214	Start logic 1 interrupt wake-up enable register	0	Table 34

Table 5. Register overview: System configuration (base address 0x4004 8000) *…continued*

Name	Access	Offset	Description	Reset value	Reference
PDSLEEPCFG	R/W	0x230	Power-down states in deep-sleep mode	0xFFFF	Table 35
PDAWAKECFG	R/W	0x234	Power-down states for wake-up from deep-sleep	0xEDF0	Table 36
PDRUNCFG	R/W	0x238	Power configuration register	0xEDF0	Table 37
DEVICE_ID	R	0x3F8	Device ID	part dependent	Table 38

4.6.1 System memory remap register

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

Table 6. System memory remap register (SYSMEMREMAP, address 0x4004 8000) bit description

Bit	Symbol	Value	Description	Reset value
1:0	MAP		System memory remap. Value 0x3 is reserved.	0x2
		0x0	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	-

4.6.2 Peripheral reset control register

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

Table 7. Peripheral reset control register (PRESETCTRL, address 0x4004 8004) bit description

Bit	Symbol	Value	Description	Reset value
0	SPI0_RST_N		SPI0 reset control	1
		0	Assert the SPI0 reset.	
		1	Clear the SPI0 reset.	
1	SPI1_RST_N		SPI1 reset control	1
		0	Assert the SPI1 reset.	
		1	Clear the SPI1 reset.	
2	UARTFRG_RST_N		USART fractional baud rate generator (UARTFRG) reset control	1
		0	Assert the UARTFRG reset.	
		1	Clear the UARTFRG reset.	

Table 7. Peripheral reset control register (PRESETCTRL, address 0x4004 8004) bit description

Bit	Symbol	Value	Description	Reset value
3	USART0_RST_N		USART0 reset control	1
		0	Assert the USART0 reset.	
		1	Clear the USART0 reset.	
4	UART1_RST_N		USART1 reset control	1
		0	Assert the USART reset.	
		1	Clear the USART1 reset.	
5	UART2_RST_N		USART2 reset control	1
		0	Assert the USART2 reset.	
		1	Clear the USART2 reset.	
6	I2C_RST_N		I2C reset control	1
		0	Assert the I2C reset.	
		1	Clear the I2C reset.	
7	MRT_RST_N		Multi-rate timer (MRT) reset control	1
		0	Assert the MRT reset.	
		1	Clear the MRT reset.	
8	SCT_RST_N		SCT reset control	1
		0	Assert the SCT reset.	
		1	Clear the SCT reset.	
9	WKT_RST_N		Self wake-up timer (WKT) reset control	1
		0	Assert the WKT reset.	
		1	Clear the WKT reset.	
10	GPIO_RST_N		GPIO and GPIO pin interrupt reset control	1
		0	Assert the GPIO reset.	
		1	Clear the GPIO reset.	
11	FLASH_RST_N		Flash controller reset control	1
		0	Assert the flash controller reset.	
		1	Clear the flash controller reset.	
12	ACMP_RST_N		Analog comparator reset control	1
		0	Assert the analog comparator reset.	
		1	Clear the analog comparator controller reset.	
31:12	-	-	Reserved	-

4.6.3 System PLL control register

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

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

Table 8. System PLL control register (SYSPLLCTRL, address 0x4004 8008) bit description

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

4.6.4 System PLL status register

This register is a Read-only register and supplies the PLL lock status (see Section 4.7.1.1).

Table 9. System PLL status register (SYSPLLSTAT, address 0x4004 800C) bit description

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

4.6.5 System oscillator control register

This register configures the frequency range for the system oscillator.

Table 10. System oscillator control register (SYSOSCCTRL, address 0x4004 8020) bit description

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

4.6.6 Watchdog oscillator control register

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

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

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

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

Table 11. Watchdog oscillator control register (WDTOSCCTRL, address 0x4004 8024) bit description

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

4.6.7 System reset status register

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

The reset value given in Table 12 applies to the POR reset.

Table 12. System reset status register (SYSRSTSTAT, address 0x4004 8030) bit description

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

4.6.8 System PLL clock source select register

This register selects the clock source for the system PLL. The SYSPLLCLKUEN register (see Section 4.6.9) must be toggled from LOW to HIGH for the update to take effect.

Table 13. System PLL clock source select register (SYSPLLCLKSEL, address 0x4004 8040) bit description

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

4.6.9 System PLL clock source update register

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

Table 14. System PLL clock source update enable register (SYSPLLCLKUEN, address 0x4004 8044) bit description

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

4.6.10 Main clock source select register

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

Bit 0 of the MAINCLKUEN register (see Section 4.6.11) must be toggled from 0 to 1 for the update to take effect.

Table 15. Main clock source select register (MAINCLKSEL, address 0x4004 8070) bit description

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

4.6.11 Main clock source update enable register

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

Table 16. Main clock source update enable register (MAINCLKUEN, address 0x4004 8074) bit description

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

4.6.12 System clock divider register

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

Table 17. System clock divider register (SYSAHBCLKDIV, address 0x4004 8078) bit description

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

4.6.13 System clock control register

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

Table 18. System clock control register (SYSAHBCLKCTRL, address 0x4004 8080) bit description

Bit	Symbol	Value	Description	Reset value
0	SYS		Enables the clock for the AHB, the APB bridge, the Cortex-M0+ core clocks, SYSCON, and the PMU. This bit is read only and always reads as 1.	1
		0	Reserved	
		1	Enable	
1	ROM		Enables clock for ROM.	1
		0	Disable	
		1	Enable	
2	RAM		Enables clock for SRAM.	1
		0	Disable	
		1	Enable	
3	FLASHREG		Enables clock for flash register interface.	1
		0	Disable	
		1	Enable	
4	FLASH		Enables clock for flash.	1
		0	Disable	
		1	Enable	
5	I2C		Enables clock for I2C.	0
		0	Disable	
		1	Enable	

Table 18. System clock control register (SYSAHBCLKCTRL, address 0x4004 8080) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
6	GPIO		Enables clock for GPIO port registers and GPIO pin interrupt registers.	0
		0	Disable	
		1	Enable	
7	SWM		Enables clock for switch matrix.	0
		0	Disable	
		1	Enable	
8	SCT		Enables clock for state configurable timer.	0
		0	Disable	
		1	Enable	
9	WKT		Enables clock for self wake-up timer.	0
		0	Disable	
		1	Enable	
10	MRT		Enables clock for multi-rate timer.	
		0	Disable	
		1	Enable	
11	SPI0		Enables clock for SPI0.	0
		0	Disable	
		1	Enable	
12	SPI1		Enables clock for SPI1.	
		0	Disable	
		1	Enable	
13	CRC		Enables clock for CRC.	0
		0	Disable	
		1	Enable	
14	UART0		Enables clock for USART0.	0
		0	Disable	
		1	Enable	
15	UART1		Enables clock for USART1.	0
		0	Disable	
		1	Enable	
16	UART2		Enables clock for USART2.	0
		0	Disable	
		1	Enable	
17	WWDT		Enables clock for WWDT.	0
		0	Disable	
		1	Enable	
18	IOCON		Enables clock for IOCON block.	0
		0	Disable	
		1	Enable	

Table 18. System clock control register (SYSAHBCLKCTRL, address 0x4004 8080) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
19	ACMP		Enables clock to analog comparator.	0
		0	Disable	
		1	Enable	
31:20	-	-	Reserved	-

4.6.14 USART clock divider register

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

Table 19. USART clock divider register (UARTCLKDIV, address 0x4004 8094) bit description

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

4.6.15 CLKOUT clock source select register

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

Bit 0 of the CLKOUTUEN register (see Section 4.6.16) must be toggled from 0 to 1 for the update to take effect.

Table 20. CLKOUT clock source select register (CLKOUTSEL, address 0x4004 80E0) bit description

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

4.6.16 CLKOUT clock source update enable register

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

Table 21. CLKOUT clock source update enable register (CLKOUTUEN, address 0x4004 80E4) bit description

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

4.6.17 CLKOUT clock divider register

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

Table 22. CLKOUT clock divider registers (CLKOUTDIV, address 0x4004 80E8) bit description

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

4.6.18 USART fractional generator divider value register

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

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

UARTCLKDIV is the USART clock configured in the UARTCLKDIV register.

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

1. The DIV value programmed in this register is the denominator of the divider used by the fractional rate generator to create the fractional component of U_PCLK.
2. The MULT value of the fractional divider is programmed in the UARTFRGMULT register. See Table 24.

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

See also:

Section 15.3.1 “Configure the USART clock and baud rate”

Section 15.7.1 “Clocking and Baud rates”

Table 23. USART fractional generator divider value register (UARTFRGDIV, address 0x4004 80F0) bit description

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

4.6.19 USART fractional generator multiplier value register

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

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

UARTCLKDIV is the USART clock configured in the UARTCLKDIV register.

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

1. The DIV denominator of the fractional divider value is programmed in the UARTFRGDIV register. See Table 23.
2. The MULT value programmed in this register is the numerator of the fractional divider value used by the fractional rate generator to create the fractional component to the baud rate.

See also:

Section 15.3.1 “Configure the USART clock and baud rate”

Section 15.7.1 “Clocking and Baud rates”

Table 24. USART fractional generator multiplier value register (UARTFRGMULT, address 0x4004 80F4) bit description

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

4.6.20 External trace buffer command register

This register works in conjunction with the MTB master register to start and stop tracing. Also see Section 26.5.4.

Table 25. External trace buffer command register (EXTTRACECMD, address 0x4004 80FC) bit description

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

4.6.21 POR captured PIO status register 0

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

Table 26. POR captured PIO status register 0 (PIOPORCAP0, address 0x4004 8100) bit description

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

4.6.22 IOCON glitch filter clock divider registers 6 to 0

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

Table 27. IOCON glitch filter clock divider registers 6 to 0 (IOCONCLKDIV[6:0], address 0x4004 8134 (IOCONCLKDIV6) to 0x004 814C (IOCONFILTCLKDIV0)) bit description

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

4.6.23 BOD control register

The BOD control register selects four separate threshold values for sending a BOD interrupt to the NVIC and for forced reset. Reset and interrupt threshold values listed in Table 28 are typical values.

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

See the LPC800 data sheet for the BOD reset and interrupt levels.

Table 28. BOD control register (BODCTRL, address 0x4004 8150) bit description

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

4.6.24 System tick counter calibration register

This register determines the value of the SYST_CALIB register.

Table 29. System tick timer calibration register (SYSTCKCAL, address 0x4004 8154) bit description

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

4.6.25 IRQ latency register

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

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

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

The default setting for this register is 0x010.

Table 30. IRQ latency register (IRQLATENCY, address 0x4004 8170) bit description

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

4.6.26 NMI source selection register

The NMI source selection register selects a peripheral interrupt as source for the NMI interrupt of the ARM Cortex-M0+ core. For a list of all peripheral interrupts and their IRQ numbers see Table 3. For a description of the NMI functionality, see Section 3.3.2.

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

Table 31. NMI source selection register (NMISRC, address 0x4004 8174) bit description

Bit	Symbol	Description	Reset value
4:0	IRQN	The IRQ number of the interrupt that acts as the Non-Maskable Interrupt (NMI) if bit 31 is 1. See Table 3 for the list of interrupt sources and their IRQ numbers.	0
30:5	-	Reserved	-
31	NMIEN	Write a 1 to this bit to enable the Non-Maskable Interrupt (NMI) source selected by bits 4:0.	0

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

4.6.27 Pin interrupt select registers

Each of these 8 registers selects one pin from all digital pins as the source of a pin interrupt or as the input to the pattern match engine. To select a pin for any of the eight pin interrupts or pattern match engine inputs, write the GPIO port pin number as 0 to 17 for pins PIO0_0 to PIO0_17 to the INTPIN bits. For example, setting INTPIN to 0x5 in PINTSEL0 selects pin PIO0_5 for pin interrupt 0.

To determine the GPIO port pin number on a given LPC800 package, see the pin description table in the data sheet.

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

Each of the 8 pin interrupts must be enabled in the NVIC using interrupt slots # 24 to 31 (see Table 3).

To use the selected pins for pin interrupts or the pattern match engine, see Section 8.5.2 “Pattern match engine”.

Table 32. Pin interrupt select registers (PINTSEL[0:7], address 0x4004 8178 (PINTSEL0) to 0x4004 8194 (PINTSEL7)) bit description

Bit	Symbol	Description	Reset value
5:0	INTPIN	Pin number select for pin interrupt or pattern match engine input. (PIO0_0 to PIO0_17 correspond to numbers 0 to 17).	0
31:6	-	Reserved	-

4.6.28 Start logic 0 pin wake-up enable register

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

Remark: Also enable the corresponding interrupts in the NVIC. See Table 3 “Connection of interrupt sources to the NVIC”.

Table 33. Start logic 0 pin wake-up enable register 0 (STARTERP0, address 0x4004 8204) bit description

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

4.6.29 Start logic 1 interrupt wake-up enable register

This register selects which interrupts wake the LPC800 from deep-sleep and power-down modes.

Remark: Also enable the corresponding interrupts in the NVIC. See Table 3 “Connection of interrupt sources to the NVIC”.

Table 34. Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description

Bit	Symbol	Value	Description	Reset value
0	SPI0		SPI0 interrupt wake-up	0
		0	Disabled	
		1	Enabled	
1	SPI1		SPI1 interrupt wake-up	0
		0	Disabled	
		1	Enabled	
2	-		Reserved	-
3	USART0		USART0 interrupt wake-up. Configure USART in synchronous slave mode.	0
		0	Disabled	
		1	Enabled	
4	USART1		USART1 interrupt wake-up. Configure USART in synchronous slave mode.	0
		0	Disabled	
		1	Enabled	
5	USART2		USART2 interrupt wake-up. Configure USART in synchronous slave mode.	0
		0	Disabled	
		1	Enabled	
7:6	-		Reserved	-
8	I2C		I2C interrupt wake-up.	0
		0	Disabled	
		1	Enabled	
11:9	-		Reserved	-
12	WWDT		WWDT interrupt wake-up	0
		0	Disabled	
		1	Enabled	
13	BOD		BOD interrupt wake-up	0
		0	Disabled	
		1	Enabled	
14	-		Reserved	-

Table 34. Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
15	WKT		Self wake-up timer interrupt wake-up	0
		0	Disabled	
		1	Enabled	
31:16			Reserved.	-

4.6.30 Deep-sleep mode configuration register

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

Remark: Hardware forces the analog blocks to be powered down in Deep-sleep and Power-down modes. An exception are the BOD and watchdog oscillator, which can be configured to remain running through this register. The WDTOSC_PD value written to the PDSLEEPCFG register is overwritten if the LOCK bit in the WWDT MOD register (see Table 143) is set. See Section 12.5.3 for details.

Table 35. Deep-sleep configuration register (PDSLEEPCFG, address 0x4004 8230) bit description

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

4.6.31 Wake-up configuration register

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

Table 36. 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	-		Reserved.	1
5	SYSOSC_PD		Crystal oscillator wake-up configuration	1
		0	Powered	
		1	Powered down	
6	WDTOSC_PD		Watchdog oscillator wake-up configuration. Changing this bit to powered-down has no effect when the LOCK bit in the WWDT MOD register is set. In this case, the watchdog oscillator is always running.	1
		0	Powered	
		1	Powered down	
7	SYSPLL_PD		System PLL wake-up configuration	1
		0	Powered	
		1	Powered down	
11:8	-		Reserved. Always write these bits as 0b1101	0b1101
14:12	-		Reserved. Always write these bits as 0b110	0b110
15	ACMP		Analog comparator wake-up configuration	1
		0	Powered	
		1	Powered down	
31:16	-	-	Reserved	0

4.6.32 Power configuration register

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

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

Table 37. Power configuration register (PDRUNCFG, address 0x4004 8238) bit description

Bit	Symbol	Value	Description	Reset value
0	IRCOUT_PD		IRC oscillator output power	0
		0	Powered	
		1	Powered down	
1	IRC_PD		IRC oscillator power down	0
		0	Powered	
		1	Powered down	
2	FLASH_PD		Flash power down	0
		0	Powered	
		1	Powered down	
3	BOD_PD		BOD power down	0
		0	Powered	
		1	Powered down	
4	-		Reserved.	1
5	SYSOSC_PD		Crystal oscillator power down	1
		0	Powered	
		1	Powered down	
6	WDTOSC_PD		Watchdog oscillator power down. Changing this bit to powered-down has no effect when the LOCK bit in the WWDT MOD register is set. In this case, the watchdog oscillator is always running.	1
		0	Powered	
		1	Powered down	
7	SYSPLL_PD		System PLL power down	1
		0	Powered	
		1	Powered down	
11:8	-		Reserved. Always write these bits as 0b1101	0b1101
14:12	-		Reserved. Always write these bits as 0b110	0b110
15	ACMP		Analog comparator power down	1
		0	Powered	
		1	Powered down	
31:16	-	-	Reserved	0

4.6.33 Device ID register

This device ID register is a read-only register and contains the part ID for each LPC800 part. This register is also read by the ISP/IAP commands (see Table 236).

Table 38. Device ID register (DEVICE_ID, address 0x4004 83F8) bit description

Bit	Symbol	Description	Reset value
31:0	DEVICEID	0x0000 8100 = LPC810M021FN8	part-dependent
		0x0000 8110 = LPC811M001FDH16	
		0x0000 8120 = LPC812M101FDH16	
		0x0000 8121 = LPC812M101FD20	
		0x0000 8122 = LPC812M101FDH20	

4.7 Functional description

4.7.1 System PLL functional description

The LPC800 uses the system PLL to create the clocks for the core and peripherals.

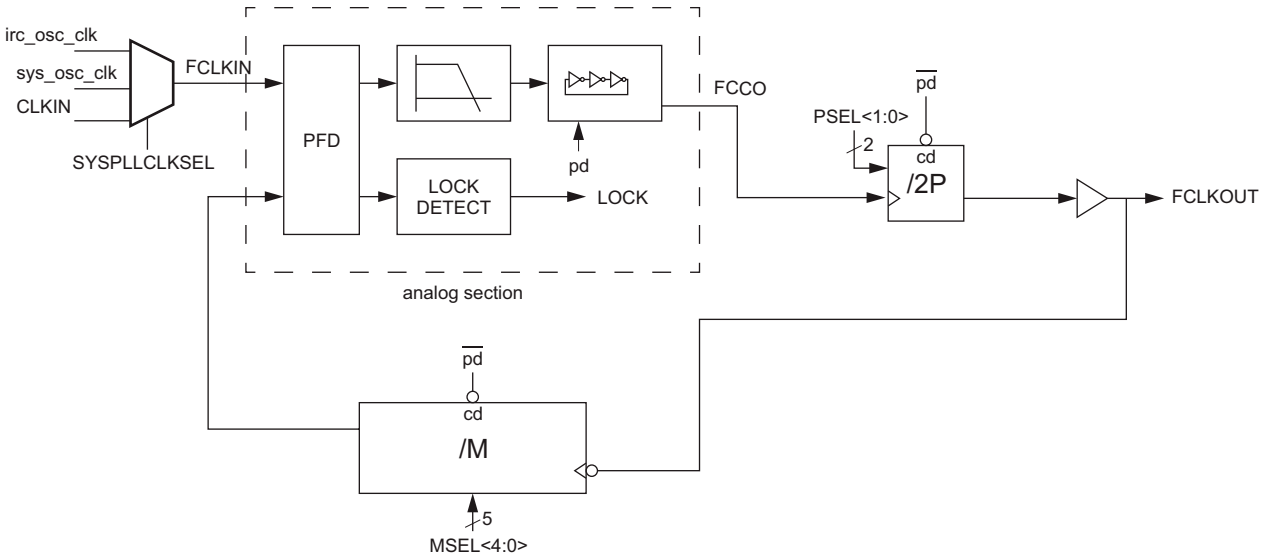


Fig 4. System PLL block diagram

The block diagram of this PLL is shown in Figure 4. The input frequency range is 10 MHz to 25 MHz. The input clock is fed directly to the Phase-Frequency Detector (PFD). This block compares the phase and frequency of its inputs, and generates a control signal when phase and/ or frequency do not match. The loop filter filters these control signals and drives the current controlled oscillator (CCO), which generates the main clock and optionally two additional phases. The CCO frequency range is 156 MHz to 320 MHz. These clocks are either divided by 2×P by the programmable post divider to create the output clocks, or are sent directly to the outputs. The main output clock is then divided by M by the programmable feedback divider to generate the feedback clock. The output signal of the phase-frequency detector is also monitored by the lock detector, to signal when the PLL has locked on to the input clock.

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

4.7.1.1 Lock detector

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

4.7.1.2 Power-down control

To reduce the power consumption when the PLL clock is not needed, a PLL Power-down mode has been incorporated. This mode is enabled by setting the SYSPLL_PD bit to one in the Power-down configuration register (Table 37). In this mode, the internal current reference will be turned off, the oscillator and the phase-frequency detector will be stopped and the dividers will enter a reset state. While in PLL Power-down mode, the lock output will be low to indicate that the PLL is not in lock. When the PLL Power-down mode is terminated by setting the SYSPLL_PD bit to zero, the PLL will resume its normal operation and will make the lock signal high once it has regained lock on the input clock.

4.7.1.3 Divider ratio programming

4.7.1.3.1 Post divider

The division ratio of the post divider is controlled by the PSEL bits. The division ratio is two times the value of P selected by PSEL bits as shown in Table 8. This guarantees an output clock with a 50% duty cycle.

4.7.1.3.2 Feedback divider

The feedback divider’s division ratio is controlled by the MSEL bits. The division ratio between the PLL’s output clock and the input clock is the decimal value on MSEL bits plus one, as specified in Table 8 .

4.7.1.3.3 Changing the divider values

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

4.7.1.4 Frequency selection

The PLL frequency equations use the following parameters (also see Figure 4):

Table 39. PLL frequency parameters

Parameter	System PLL
FCLKIN	Frequency of sys_pllclkin (input clock to the system PLL) from the SYSPLLCLKSEL multiplexer (see Section 4.6.8).
FCCO	Frequency of the Current Controlled Oscillator (CCO); 156 to 320 MHz.
FCLKOUT	Frequency of sys_pllclkout. This is the PLL output frequency and must be < 100 MHz.
P	System PLL post divider ratio; PSEL bits in SYSPLLCTRL (see Section 4.6.3).
M	System PLL feedback divider register; MSEL bits in SYSPLLCTRL (see Section 4.6.3).

4.7.1.4.1 Normal mode

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

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

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

1. Specify the input clock frequency Fclkin.
2. Calculate M to obtain the desired output frequency Fclkout with M = F_{clkout} / F_{clkin}.
3. Find a value so that FCCO = 2 × P × F_{clkout}.
4. Verify that all frequencies and divider values conform to the limits specified in Table 8.

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

Table 40 shows how to configure the PLL for a 12 MHz crystal oscillator using the SYSPLLCTRL register (Table 8). The main clock is equivalent to the system clock if the system clock divider SYSAHBCLKDIV is set to one (see Table 17).

Table 40. PLL configuration examples

PLL input clock sys_pllclkin (Fclkin)	Main clock (Fclkout)	MSEL bits Table 8	M divider value	PSEL bits Table 8	P divider value	FCCO frequency	SYSAHBCLKDIV	System clock
12 MHz	60 MHz	00100 (binary)	5	01 (binary)	2	240 MHz	2	30 MHz
12 MHz	24 MHz	00001(binary)	2	10 (binary)	4	192 MHz	1	24 MHz

4.7.1.4.2 PLL Power-down mode

In this mode, the internal current reference will be turned off, the oscillator and the phase-frequency detector will be stopped and the dividers will enter a reset state. While in PLL Power-down mode, the lock output will be low, to indicate that the PLL is not in lock. When the PLL Power-down mode is terminated by SYSPLL_PD bit to zero in the Power-down configuration register (Table 37), the PLL will resume its normal operation and will make the lock signal high once it has regained lock on the input clock.

5.1 How to read this chapter

The LPC800 provides an on-chip API in the boot ROM to optimize power consumption in active and sleep modes. See Table 255 “Power profile API calls”.

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

5.2 Features

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

5.3 Basic configuration

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

If the open-drain pins PIO0_10 and PIO0_11 are not pinned out, you must enable their output driver and drive the outputs internally LOW to minimize power consumption in the low power modes. See Section 6.3.

5.3.1 Low power modes in the ARM Cortex-M0+ core

Entering and exiting the low power modes is always controlled by the ARM Cortex-M0+ core. The SCR register is the software interface for controlling the core’s actions when entering a low power mode. The SCR register is located on the ARM private peripheral bus. For details, see Ref. 1.

5.3.1.1 System control register

The System control register (SCR) controls entry to and exit from a low power state. This register is a R/W register with reset value of 0x0000 0000. The SCR register allows to put the ARM core into sleep mode or the entire system in Deep-sleep or Power-down mode. To set the low power state with SLEEPDEEP = 1 to either deep-sleep or power-down or to enter the Deep power-down mode, use the PCON register (Table 44).

Table 41. System control register (SCR, address 0xE000 ED10) bit description

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

5.4 Pin description

In Deep power-down only the WAKEUP pin (pin PIO0_4) is functional. The WAKEUP function can be disabled in the DPDCTRL register to lower the power consumption even more. In this case, enable the self wake-up timer to provide an internal wake-up signal. See Section 5.6.3 “Deep power-down control register”.

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

5.5 General description

Power on the LPC800 is controlled by the PMU, by the SYSCON block, and the ARM Cortex-M0+ core. The following reduced power modes are supported in order from highest to lowest power consumption:

1. Sleep mode:

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

2. Deep-sleep and power-down modes:

   The Deep-sleep and power-down modes affect the core and the entire system with memories and peripherals.

   a. In Deep-sleep mode, the peripherals receive no internal clocks. The flash is in stand-by mode. The SRAM memory and all peripheral registers as well as the processor maintain their internal states. The WWDT, WKT, and BOD can remain active to wake up the system on an interrupt.

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

3. Deep power-down mode:

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

Remark: The LPC800 is in active mode when it is fully powered and operational after booting.

5.5.1 Wake-up process

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

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

- To configure the system after wake-up: Table 36 “Wake-up configuration register (PDAWAKECFG, address 0x4004 8234) bit description”.
- To use external interrupts for wake-up: Table 33 “Start logic 0 pin wake-up enable register 0 (STARTERP0, address 0x4004 8204) bit description” and Table 32 “Pin interrupt select registers (PINTSEL[0:7], address 0x4004 8178 (PINTSEL0) to 0x4004 8194 (PINTSEL7)) bit description”
- To enable external or internal signals to wake up the part from Deep-sleep or Power-down modes: Table 34 “Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description”
- To configure the USART to wake up the part: Section 15.3.2 “Configure the USART for wake-up”
- For configuring the self wake-up timer: Section 13.5

- For a list of all wake-up sources: Table 42 “Wake-up sources for reduced power modes”

Table 42. Wake-up sources for reduced power modes

Power mode	Wake-up source	Conditions
Sleep	Any interrupt	Enable interrupt in NVIC.
Deep-sleep and Power-down	Pin interrupts	Enable pin interrupts in NVIC and STARTERP0 registers.
	BOD interrupt	• Enable interrupt in NVIC and STARTERP1 registers. • Enable interrupt in BODCTRL register. • BOD powered in PDSLEEPCFG register.
	BOD reset	• Enable reset in BODCTRL register. • BOD powered in PDSLEEPCFG register.
	WWDT interrupt	• Enable interrupt in NVIC and STARTERP1 registers. • WWDT running. Enable WWDT in WWDT MOD register and feed. • Enable interrupt in WWDT MOD register. • WDOsc powered in PDSLEEPCFG register.
	WWDT reset	• WWDT running. • Enable reset in WWDT MOD register. • WDOsc powered in PDSLEEPCFG register.
	Self Wake-up Timer (WKT) time-out	• Enable interrupt in NVIC and STARTERP1 registers. • Enable low-power oscillator in the DPDCTRL register in the PCON block. • Select low-power clock for WKT clock in the WKT CTRL register. • Start the WKT by writing a time-out value to the WKT COUNT register.
	Interrupt from USART/SPI/I2C peripheral	• Enable interrupt in NVIC and STARTERP1 registers. • Enable USART/I2C/SPI interrupts. • Provide an external clock signal to the peripheral. • Configure the USART in synchronous slave mode and I2C and SPI in slave mode.
Deep power-down	WAKEUP pin PIO0_4	Enable the WAKEUP function in the DPDCTRL register in the PMU.
	WKT time-out	• Enable the low-power oscillator in the DPDCTRL register in the PMU. • Enable the low-power oscillator to keep running in Deep power-down mode in the DPDCTRL register in the PMU. • Select low-power clock for WKT clock in the WKT CTRL register. • Start WKT by writing a time-out value to the WKT COUNT register.

5.6 Register description

Table 43. Register overview: PMU (base address 0x4002 0000)

Name	Access	Address offset	Description	Reset value	Reference
PCON	R/W	0x000	Power control register	0x0	Table 44
GPREG0	R/W	0x004	General purpose register 0	0x0	Table 45
GPREG1	R/W	0x008	General purpose register 1	0x0	Table 45

Table 43. Register overview: PMU (base address 0x4002 0000) *…continued*

Name	Access	Address offset	Description	Reset value	Reference
GPREG2	R/W	0x00C	General purpose register 2	0x0	Table 45
GPREG3	R/W	0x010	General purpose register 3	0x0	Table 45
DPDCTRL	R/W	0x014	Deep power-down control register	0x0	Table 46

5.6.1 Power control register

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

Table 44. Power control register (PCON, address 0x4002 0000) bit description

Bit	Symbol	Value	Description	Reset value
2:0	PM		Power mode	000
		0x0	Default. The part is in active or sleep mode.	
		0x1	ARM WFI will enter Deep-sleep mode.	
		0x2	ARM WFI will enter Power-down mode.	
		0x3	ARM WFI will enter Deep-power down mode (ARM Cortex-M0+ core powered-down).	
3	NODPD		A 1 in this bit prevents entry to Deep power-down mode when 0x3 is written to the PM field above, the SLEEPDEEP bit is set, and a WFI is executed. This bit is cleared only by power-on reset, so writing a one to this bit locks the part in a mode in which Deep power-down mode is blocked.	0
7:4	-	-	Reserved. Do not write ones to this bit.	0
8	SLEEPFLAG		Sleep mode flag	0
		0	Read: No power-down mode entered. Part is in Active 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.	0
11	DPDFLAG		Deep power-down flag	0
		0	Read: Deep power-down mode **not** entered. Write: No effect.	0
		1	Read: Deep power-down mode entered. Write: Clear the Deep power-down flag.	
31:12	-	-	Reserved. Do not write ones to this bit.	0

5.6.2 General purpose registers 0 to 3

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

Table 45. General purpose registers 0 to 3 (GPREG[0:3], address 0x4002 0004 (GPREG0) to 0x4002 0010 (GPREG3)) bit description

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

5.6.3 Deep power-down control register

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

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

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

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

Table 46. Deep power down control register (DPDCTRL, address 0x4002 0014) bit description

Bit	Symbol	Value	Description	Reset value
0	WAKEUPHYS		WAKEUP pin hysteresis enable	0
		0	Disabled. Hysteresis for WAKEUP pin disabled.	
		1	Enabled. Hysteresis for WAKEUP pin enabled.	
1	WAKEPAD_ DISABLE		WAKEUP pin disable. Setting this bit disables the wake-up pin, so it can be used for other purposes. **Remark:** Never set this bit if you intend to use a pin to wake up the part from Deep power-down mode. You can only disable the wake-up pin if the self wake-up timer is enabled and configured. **Remark:** Setting this bit is not necessary if Deep power-down mode is not used.	0
		0	Enabled. The wake-up function is enabled on pin PIO0_4.	
		1	Disabled. Setting this bit disables the wake-up function on pin PIO0_4.	

Table 46. Deep power down control register (DPDCTRL, address 0x4002 0014) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
2	LPOSCEN		Enable the low-power oscillator for use with the 10 kHz self wake-up timer clock. You must set this bit if the CLKSEL bit in the self wake-up timer CTRL bit is set. Do not enable the low-power oscillator if the self wake-up timer is clocked by the divided IRC.	0
		0	Disabled.	
		1	Enabled.	
3	LPOSCDPDEN		Enable the low-power oscillator in Deep power-down mode. Setting this bit causes the low-power oscillator to remain running during Deep power-down mode provided that bit 2 in this register is set as well. You must set this bit for the self wake-up timer to be able to wake up the part from Deep power-down mode. **Remark:** Do not set this bit unless you use the self wake-up timer to wake up from Deep power-down mode.	0
		0	Disabled.	
		1	Enabled.	
31:4	-		Data retained during Deep power-down mode.	0x0

5.7 Functional description

5.7.1 Power management

The LPC800 support a variety of power control features. In Active mode, when the chip is running, power and clocks to selected peripherals can be optimized for power consumption. In addition, there are four special modes of processor power reduction with different peripherals running: Sleep mode, Deep-sleep mode, Power-down mode, and Deep power-down mode.

Table 47. Peripheral configuration in reduced power modes

Peripheral	Sleep mode	Deep-sleep mode	Power-down mode	Deep power-down mode
IRC	software configurable	on	off	off
IRC output	software configurable	off	off	off
Flash	software configurable	on	off	off
BOD	software configurable	software configurable	software configurable	off
PLL	software configurable	off	off	off
SysOsc	software configurable	off	off	off
WDosc/WWDT	software configurable	software configurable	software configurable	off
Digital peripherals	software configurable	off	off	off
WKT/low-power oscillator	software configurable	software configurable	software configurable	software configurable

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

5.7.2 Reduced power modes and WWDT lock features

The WWDT lock feature influences the power consumption in any of the power modes because locking the WWDT clock source forces the watchdog oscillator to be on independently of the Deep-sleep and Power-down mode software configuration through the PDSLEEPCFG register. For details see Section 12.5.3 “Using the WWDT lock features”.

5.7.3 Active mode

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

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

5.7.3.1 Power configuration in Active mode

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

- The SYSAHBCLKCTRL register controls which memories and peripherals are running (Table 18).
- The power to various analog blocks (PLL, oscillators, the BOD circuit, and the flash block) can be controlled at any time individually through the PDRUNCFG register (Table 37 “Power configuration register (PDRUNCFG, address 0x4004 8238) bit description”).
- The clock source for the system clock can be selected from the IRC (default), the system oscillator, or the watchdog oscillator (see Figure 3 and related registers).
- The system clock frequency can be selected by the SYSPLLCTRL (Table 8) and the SYSAHBCLKDIV register (Table 17).
- The USART and CLKOUT use individual peripheral clocks with their own clock dividers. The peripheral clocks can be shut down through the corresponding clock divider registers.

5.7.4 Sleep mode

In Sleep mode, the system clock to the ARM Cortex-M0+ core is stopped and execution of instructions is suspended until either a reset or an interrupt occurs.

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

5.7.4.1 Power configuration in Sleep mode

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

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

5.7.4.2 Programming Sleep mode

The following steps must be performed to enter Sleep mode:

1. The PM bits in the PCON register must be set to the default value 0x0.
2. The SLEEPDEEP bit in the ARM Cortex-M0+ SCR register must be set to zero (Table 41).
3. Use the ARM Cortex-M0+ Wait-For-Interrupt (WFI) instruction.

5.7.4.3 Wake-up from Sleep mode

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

5.7.5 Deep-sleep mode

In Deep-sleep mode, the system clock to the processor is disabled as in Sleep mode. All analog blocks are powered down, except for the BOD circuit and the watchdog oscillator, which can be selected or deselected during Deep-sleep mode in the PDSLEEPCFG register. The main clock, and therefore all peripheral clocks, are disabled except for the clock to the watchdog timer if the watchdog oscillator is selected. The IRC is running, but its output is disabled. The flash is in stand-by mode.

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

5.7.5.1 Power configuration in Deep-sleep mode

Power consumption in Deep-sleep mode is determined by the Deep-sleep power configuration setting in the PDSLEEPCFG (Table 35) register:

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

5.7.5.2 Programming Deep-sleep mode

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

1. The PM bits in the PCON register must be set to 0x1 (Table 44).
2. Select the power configuration in Deep-sleep mode in the PDSLEEPCFG (Table 35) register.

3. Select the power configuration after wake-up in the PDAWAKECFG (Table 36) register.
4. If any of the available wake-up interrupts are needed for wake-up, enable the interrupts in the interrupt wake-up registers (Table 33, Table 34) and in the NVIC.
5. Write one to the SLEEPDEEP bit in the ARM Cortex-M0+ SCR register (Table 41).
6. Use the ARM WFI instruction.

5.7.5.3 Wake-up from Deep-sleep mode

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

- Signal on one of the eight pin interrupts selected in Table 32. Each pin interrupt must also be enabled in the STARTERP0 register (Table 33) and in the NVIC.
- BOD signal, if the BOD is enabled in the PDSLEEPCFG register:
  - BOD interrupt using the deep-sleep interrupt wake-up register 1 (Table 34). The BOD interrupt must be enabled in the NVIC. The BOD interrupt must be selected in the BODCTRL register.
  - Reset from the BOD circuit. In this case, the BOD circuit must be enabled in the PDSLEEPCFG register, and the BOD reset must be enabled in the BODCTRL register (Table 28).
- WWDT signal, if the watchdog oscillator is enabled in the PDSLEEPCFG register:
  - WWDT interrupt using the interrupt wake-up register 1 (Table 34). The WWDT interrupt must be enabled in the NVIC. The WWDT interrupt must be set in the WWDT MOD register, and the WWDT must be enabled in the SYSAHBCLKCTRL register.
  - Reset from the watchdog timer. The WWDT reset must be set in the WWDT MOD register. In this case, the watchdog oscillator must be running in Deep-sleep mode (see PDSLEEPCFG register), and the WDT must be enabled in the SYSAHBCLKCTRL register.
- Via any of the USART blocks if the USART is configured in synchronous mode. See Section 15.3.2 “Configure the USART for wake-up”.
- Via the I2C. See Section 16.3.2.
- Via any of the SPI blocks. See Section 17.3.1.

Remark:

5.7.6 Power-down mode

In Power-down mode, the system clock to the processor is disabled as in Sleep mode. All analog blocks are powered down, except for the BOD circuit and the watchdog oscillator, which must be selected or deselected during Power-down mode in the PDSLEEPCFG register. The main clock and therefore all peripheral clocks are disabled except for the clock to the watchdog timer if the watchdog oscillator is selected. The IRC itself and the flash are powered down, decreasing power consumption compared to Deep-sleep mode.

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

5.7.6.1 Power configuration in Power-down mode

Power consumption in Power-down mode can be configured by the power configuration setting in the PDSLEEPCFG (Table 35) register in the same way as for Deep-sleep mode (see Section 5.7.5.1):

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

5.7.6.2 Programming Power-down mode

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

1. The PM bits in the PCON register must be set to 0x2 (Table 44).
2. Select the power configuration in Power-down mode in the PDSLEEPCFG (Table 35) register.
3. Select the power configuration after wake-up in the PDAWAKECFG (Table 36) register.
4. If any of the available wake-up interrupts are used for wake-up, enable the interrupts in the interrupt wake-up registers (Table 33, Table 34) and in the NVIC.
5. Write one to the SLEEPDEEP bit in the ARM Cortex-M0+ SCR register (Table 41).
6. Use the ARM WFI instruction.

5.7.6.3 Wake-up from Power-down mode

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

- Signal on one of the eight pin interrupts selected in Table 32. Each pin interrupt must also be enabled in the STARTERP0 register (Table 33) and in the NVIC.
- BOD signal, if the BOD is enabled in the PDSLEEPCFG register:
  - BOD interrupt using the interrupt wake-up register 1 (Table 34). The BOD interrupt must be enabled in the NVIC. The BOD interrupt must be selected in the BODCTRL register.
  - Reset from the BOD circuit. In this case, the BOD reset must be enabled in the BODCTRL register (Table 28).
- WWDT signal, if the watchdog oscillator is enabled in the PDSLEEPCFG register:
  - WWDT interrupt using the interrupt wake-up register 1 (Table 34). The WWDT interrupt must be enabled in the NVIC. The WWDT interrupt must be set in the WWDT MOD register.
  - Reset from the watchdog timer. The WWDT reset must be set in the WWDT MOD register.

- Via any of the USART blocks. See Section 15.3.2 “Configure the USART for wake-up”.
- Via the I2C. See Section 16.3.2.
- Via any of the SPI blocks. See Section 17.3.1.

5.7.7 Deep power-down mode

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

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

All functional pins are tri-stated in Deep power-down mode except for the WAKEUP pin. In this mode, you must pull the $\overline{\text{RESET}}$ pin HIGH externally.

Remark: Setting bit 3 in the PCON register (Table 44) prevents the part from entering Deep-power down mode.

5.7.7.1 Power configuration in Deep power-down mode

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

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

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

1. Pull the WAKEUP pin externally HIGH.
2. Ensure that bit 3 in the PCON register (Table 44) is cleared.
3. Write 0x3 to the PM bits in the PCON register (see Table 44).
4. Store data to be retained in the general purpose registers (Section 5.6.2).
5. Write one to the SLEEPDEEP bit in the ARM Cortex-M0+ SCR register (Table 41).
6. Use the ARM WFI instruction.

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

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

1. On the WAKEUP pin, transition from HIGH to LOW.
   - The PMU will turn on the on-chip voltage regulator. When the core voltage reaches the power-on-reset (POR) trip point, a system reset will be triggered and the chip re-boots.
   - All registers except the DPDCTRL and GPREG0 to GPREG3registers will be in their reset state.
2. Once the chip has booted, read the deep power-down flag in the PCON register (Table 44) to verify that the reset was caused by a wake-up event from Deep power-down and was not a cold reset.

3. Clear the deep power-down flag in the PCON register (Table 44).
4. (Optional) Read the stored data in the general purpose registers (Section 5.6.2).
5. Set up the PMU for the next Deep power-down cycle.

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

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

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

1. Enable the the low-power oscillator to run in Deep power-down mode by setting bits 2 and 3 in the DPDCTRL register to 1 (see Table 46)
2. Ensure that bit 3 in the PCON register (Table 44) is cleared.
3. Write 0x3 to the PM bits in the PCON register (see Table 44).
4. Store data to be retained in the general purpose registers (Section 5.6.2).
5. Write one to the SLEEPDEEP bit in the ARM Cortex-M0+ SCR register.
6. Start the self-wake-up timer by writing a value to the WKT COUNT register (Table 152).
7. Use the ARM WFI instruction.

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

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

1. When the WKT count reaches 0, the following happens:
   - The PMU will turn on the on-chip voltage regulator. When the core voltage reaches the power-on-reset (POR) trip point, a system reset will be triggered and the chip re-boots.
   - All registers except the DPDCTRL and GPREG0 to GPREG3 registers will be in their reset state.
2. Once the chip has booted, read the deep power-down flag in the PCON register (Table 44) to verify that the reset was caused by a wake-up event from Deep power-down and was not a cold reset.
3. Clear the deep power-down flag in the PCON register (Table 44).
4. (Optional) Read the stored data in the general purpose registers (Section 5.6.2).
5. Set up the PMU for the next Deep power-down cycle.

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

6.1 How to read this chapter

The IOCON block is identical for all LPC800 parts. Registers for pins that are not available on a specific package are reserved.

Table 48. Pinout summary

Package	Pins/configuration registers available
TSSOP16	PIO0_0 to PIO0_13
TSSOP20	PIO0_0 to PIO0_17
SOP20	PIO0_0 to PIO0_17
DIP8	PIO0_0 to PIO0_5

6.2 Features

The following electrical properties are configurable for each pin:

- Pull-up/pull-down resistor
- Open-drain mode
- Hysteresis
- Digital glitch filter with programmable time constant
- Analog mode (for a subset of pins, see the LPC81xM data sheet)

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

6.3 Basic configuration

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

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

6.4 General description

6.4.1 Pin configuration

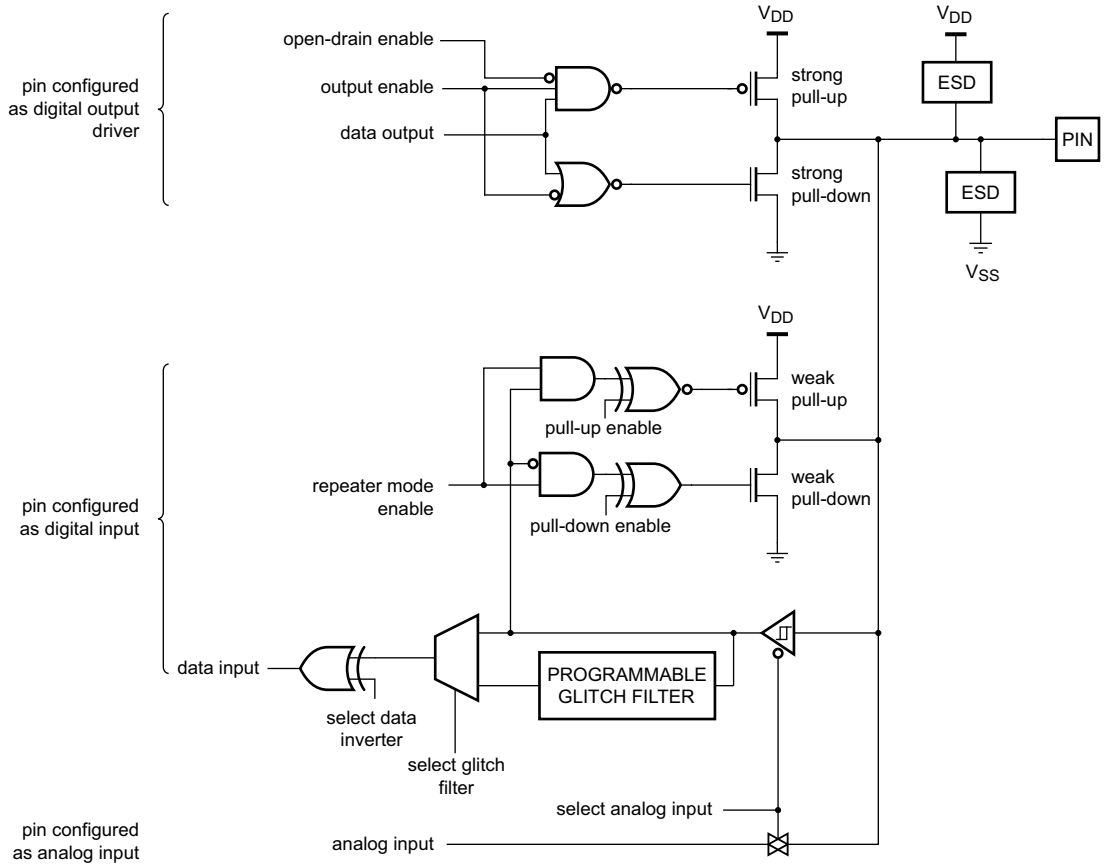


Fig 5. Pin configuration

6.4.2 Pin function

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

Related links:

Table 95 “Movable functions (assign to pins PIO0_0 to PIO0_17 through switch matrix)”

6.4.3 Pin mode

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

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

6.4.4 Open-drain mode

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

6.4.5 Analog mode

The switch matrix automatically configures the pin in analog mode whenever an analog input or output is selected as the pin's function.

6.4.6 I^2C-bus mode

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

6.4.7 Programmable glitch filter

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

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

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

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

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

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

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

Related registers and links:

Table 27 “IOCON glitch filter clock divider registers 6 to 0 (IOCONCLKDIV[6:0], address 0x4004 8134 (IOCONCLKDIV6) to 0x004 814C (IOCONFILTCLKDIV0)) bit description”

6.5 Register description

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

Table 49. Register overview: I/O configuration (base address 0x4004 4000)

Name	Access	Address offset	Description	Reset value	Reference
PIO0_17	R/W	0x000	I/O configuration for pin PIO0_17	0x0000 0090	Table 50
PIO0_13	R/W	0x004	I/O configuration for pin PIO0_13	0x0000 0090	Table 51
PIO0_12	R/W	0x008	I/O configuration for pin PIO0_12	0x0000 0090	Table 52
PIO0_5	R/W	0x00C	I/O configuration for pin PIO0_5/$\overline{\text{RESET}}$	0x0000 0090	Table 53
PIO0_4	R/W	0x010	I/O configuration for pin PIO0_4	0x0000 0090	Table 54
PIO0_3	R/W	0x014	I/O configuration for pin PIO0_3/SWCLK	0x0000 0090	Table 55
PIO0_2	R/W	0x018	I/O configuration for pin PIO0_2/SWDIO	0x0000 0090	Table 56
PIO0_11	R/W	0x01C	I/O configuration for pin PIO0_11. This is the pin configuration for the true open-drain pin.	0x0000 0080	Table 57
PIO0_10	R/W	0x020	I/O configuration for pin PIO0_10. This is the pin configuration for the true open-drain pin.	0x0000 0080	Table 58
PIO0_16	R/W	0x024	I/O configuration for pin PIO0_16	0x0000 0090	Table 59
PIO0_15	R/W	0x028	I/O configuration for pin PIO0_15	0x0000 0090	Table 60
PIO0_1	R/W	0x02C	I/O configuration for pin PIO0_1/ACMP_I1/CLKIN	0x0000 0090	Table 61
-	-	0x030	Reserved	-	-
PIO0_9	R/W	0x034	I/O configuration for pin PIO0_9/XTALOUT	0x0000 0090	Table 62
PIO0_8	R/W	0x038	I/O configuration for pin PIO0_8/XTALIN	0x0000 0090	Table 63
PIO0_7	R/W	0x03C	I/O configuration for pin PIO0_7	0x0000 0090	Table 64
PIO0_6	R/W	0x040	I/O configuration for pin PIO0_6/VDDCMP	0x0000 0090	Table 65
PIO0_0	R/W	0x044	I/O configuration for pin PIO0_0/ACMP_I0	0x0000 0090	Table 66
PIO0_14	R/W	0x048	I/O configuration for pin PIO0_14	0x0000 0090	Table 67

6.5.1 PIO0_17 register

Table 50. PIO0_17 register (PIO0_17, address 0x4004 4000) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0b10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.2 PIO0_13 register

Table 51. PIO0_13 register (PIO0_13, address 0x4004 4004) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0b10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.3 PIO0_12 register

Table 52. PIO0_12 register (PIO0_12, address 0x4004 4008) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0b10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.4 PIO0_5 register

Table 53. PIO0_5 register (PIO0_5, address 0x4004 400C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0b10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.5 PIO0_4 register

Table 54. PIO0_4 register (PIO0_4, address 0x4004 4010) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0b10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.6 PIO0_3 register

Table 55. PIO0_3 register (PIO0_3, address 0x4004 4014) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0b10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input.	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.7 PIO0_2 register

Table 56. PIO0_2 register (PIO0_2, address 0x4004 4018) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0b10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input.	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.8 PIO0_11 register

Table 57. PIO0_11 register (PIO0_11, address 0x4004 401C) bit description

Bit	Symbol	Value	Description	Reset value
5:0	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
7	-		Reserved.	1
9:8	I2CMODE		Selects I2C mode. Select Standard mode (I2CMODE = 00, default) or Standard I/O functionality (I2CMODE = 01) if the pin function is GPIO (FUNC = 000).	00
		0x0	Standard mode/ Fast-mode I2C.	
		0x1	Standard I/O functionality	
		0x2	Fast-mode Plus I2C	
		0x3	Reserved.	
10	-	-	Reserved.	-
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	-

6.5.9 PIO0_10 register

Table 58. PIO0_10 register (PIO0_10, address 0x4004 4020) bit description

Bit	Symbol	Value	Description	Reset value
5:0	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
7	-		Reserved.	1
9:8	I2CMODE		Selects I2C mode. Select Standard mode (I2CMODE = 00, default) or Standard I/O functionality (I2CMODE = 01) if the pin function is GPIO (FUNC = 000).	00
		0x0	Standard mode/ Fast-mode I2C.	
		0x1	Standard I/O functionality	
		0x2	Fast-mode Plus I2C	
		0x3	Reserved.	
10	-	-	Reserved.	-
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	-

6.5.10 PIO0_16 register

Table 59. PIO0_16 register (PIO0_16, address 0x4004 4024) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0b10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.11 PIO0_15 register

Table 60. PIO0_15 register (PIO0_15, address 0x4004 4028) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0b10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.12 PIO0_1 register

Table 61. PIO0_1 register (PIO0_1, address 0x4004 402C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0b10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.13 PIO0_9 register

Table 62. PIO0_9 register (PIO0_9, address 0x4004 4034) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.14 PIO0_8 register

Table 63. PIO0_8 register (PIO0_8, address 0x4004 4038) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0b10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.15 PIO0_7 register

Table 64. PIO0_7 register (PIO0_7, address 0x4004 403C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0b10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.16 PIO0_6 register

Table 65. PIO0_6 register (PIO0_6, address 0x4004 4040) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.17 PIO0_0 register

Table 66. PIO0_0 register (PIO0_0, address 0x4004 4044) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0b10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

6.5.18 PIO0_14 register

Table 67. PIO0_14 register (PIO0_14, address 0x4004 4048) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0b10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

7.1 How to read this chapter

All GPIO registers refer to 32 pins per port. Depending on the package type, not all pins are available, and the corresponding bits in the GPIO registers are reserved (see Table 68).

Table 68. GPIO pins available

Package	GPIO Port 0
TSSOP16	PIO0_0 to PIO0_13
TSSOP20	PIO0_0 to PIO0_17
SOP20	PIO0_0 to PIO0_17
DIP8	PIO0_0 to PIO0_5

7.2 Features

- GPIO port registers are located on the ARM Cortex M0+ I/O port for fast access.
- The ARM Cortex M0+ I/O port supports single-cycle access.
- GPIO ports
  - GPIO pins can be configured as input or output by software.
  - All GPIO pins default to inputs at reset.
  - Pin interrupt registers allow pins to be sensed and set individually.

7.3 Basic configuration

For the GPIO port registers, enable the clock to the GPIO port registers in the SYSAHBCLKCTRL register (Table 18, bit 6).

7.4 Pin description

All GPIO functions are fixed-pin functions. The switch matrix assigns every GPIO port pin to one and only one pin on the LPC800 package. By default, the switch matrix connects all package pins except supply and ground pins to their GPIO port pins.

The pin description table (see Table 291) shows how the GPIO port pins are assigned to LPC800 package pins.

7.5 General description

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

7.6 Register description

The GPIO port registers and the GPIO pin interrupt registers are located on the ARM M0+ I/O port. The I/O port supports single-cycle access.

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

“ext” indicates that the data read after reset depends on the state of the pin, which in turn may depend on an external source.

Remark: You can program reserved bits in the GPIO registers to prevent the open-drain I2C pins from internally floating when not pinned out. See Section 6.3.

Table 69. Register overview: GPIO port (base address 0xA000 0000)

Name	Access	Address offset	Description	Reset value	Width	Reference
B0 to B17	R/W	0x0000 to 0x0012	Byte pin registers port 0; pins PIO0_0 to PIO0_17	ext	byte (8 bit)	Table 70
W0 to W17	R/W	0x1000 to 0x1048	Word pin registers port 0	ext	word (32 bit)	Table 71
DIR0	R/W	0x2000	Direction registers port 0	0	word (32 bit)	Table 72
MASK0	R/W	0x2080	Mask register port 0	0	word (32 bit)	Table 73
PIN0	R/W	0x2100	Port pin register port 0	ext	word (32 bit)	Table 74
MPIN0	R/W	0x2180	Masked port register port 0	ext	word (32 bit)	Table 75
SET0	R/W	0x2200	Write: Set register for port 0 Read: output bits for port 0	0	word (32 bit)	Table 76
CLR0	WO	0x2280	Clear port 0	NA	word (32 bit)	Table 77
NOT0	WO	0x2300	Toggle port 0	NA	word (32 bit)	Table 78

7.6.1 GPIO port byte pin registers

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

Table 70. GPIO port 0 byte pin registers (B[0:17], addresses 0xA000 0000 (B0) to 0xA000 0012 (B17)) bit description

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

7.6.2 GPIO port word pin registers

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

Table 71. GPIO port 0 word pin registers (W[0:17], addresses 0xA000 1000 (W0) to 0x5000 1048 (W17)) bit description

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

7.6.3 GPIO port direction registers

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

Table 72. GPIO direction port 0 register (DIR0, address 0xA000 2000) bit description

Bit	Symbol	Description	Reset value	Access
17:0	DIRP0	Selects pin direction for pin PIO0_n (bit 0 = PIO0_0, bit 1 = PIO0_1, ..., bit 17 = PIO0_17). 0 = input. 1 = output.	0	R/W
31:18	-	Reserved.	0	-

7.6.4 GPIO port mask registers

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

Table 73. GPIO mask port 0 register (MASK0, address 0xA000 2080) bit description

Bit	Symbol	Description	Reset value	Access
17:0	MASKP0	Controls which bits corresponding to PIO0_n are active in the P0MPORT register (bit 0 = PIO0_0, bit 1 = PIO0_1, ..., bit 17 = PIO0_17). 0 = Read MPORT: pin state; write MPORT: load output bit. 1 = Read MPORT: 0; write MPORT: output bit not affected.	0	R/W
31:18	-	Reserved.	0	-

7.6.5 GPIO port pin registers

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

Table 74. GPIO port 0 pin register (PIN0, address 0xA000 2100) bit description

Bit	Symbol	Description	Reset value	Access
17:0	PORT0	Reads pin states or loads output bits (bit 0 = PIO0_0, bit 1 = PIO0_1, ..., bit 17 = PIO0_17). 0 = Read: pin is low; write: clear output bit. 1 = Read: pin is high; write: set output bit.	ext	R/W
31:18	-	Reserved.	0	-

7.6.6 GPIO masked port pin registers

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

Table 75. GPIO masked port 0 pin register (MPIN0, address 0xA000 2180) bit description

Bit	Symbol	Description	Reset value	Access
17:0	MPORTP0	Masked port register (bit 0 = PIO0_0, bit 1 = PIO0_1, ..., bit 17 = PIO0_17). 0 = Read: pin is LOW and/or the corresponding bit in the MASK register is 1; write: clear output bit if the corresponding bit in the MASK register is 0. 1 = Read: pin is HIGH and the corresponding bit in the MASK register is 0; write: set output bit if the corresponding bit in the MASK register is 0.	ext	R/W
31:18	-	Reserved.	0	-

7.6.7 GPIO port set registers

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

Table 76. GPIO set port 0 register (SET0, address 0xA000 2200) bit description

Bit	Symbol	Description	Reset value	Access
17:0	SETP0	Read or set output bits. 0 = Read: output bit: write: no operation. 1 = Read: output bit; write: set output bit.	0	R/W
31:18	-	Reserved.	0	-

7.6.8 GPIO port clear registers

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

Table 77. GPIO clear port 0 register (CLR0, address 0xA000 2280) bit description

Bit	Symbol	Description	Reset value	Access
17:0	CLRP0	Clear output bits: 0 = No operation. 1 = Clear output bit.	NA	WO
31:18	-	Reserved.	0	-

7.6.9 GPIO port toggle registers

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

Table 78. GPIO toggle port 0 register (NOT0, address 0xA000 2300) bit description

Bit	Symbol	Description	Reset value	Access
17:0	NOTP0	Toggle output bits: 0 = no operation. 1 = Toggle output bit.	NA	WO
31:18	-	Reserved.	0	-

7.7 Functional description

7.7.1 Reading pin state

Software can read the state of all GPIO pins except those selected for an analog function in the switch matrix logic. A pin does not have to be selected for GPIO in the switch matrix in order to read its state. There are several ways to read the pin state:

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

7.7.2 GPIO output

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

1. The pin must be selected for GPIO operation in the switch matrix.
2. The pin must be selected for output by a 1 in its port's DIR register.

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

There are multiple ways to change GPIO output bits:

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

The state of a port’s output bits can be read from its SET register. Reading any of the registers described in Section 7.7.1 returns the state of pins, regardless of their direction or alternate functions.

7.7.3 Masked I/O

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

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

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

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

7.7.4 Recommended practices

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

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

8.1 How to read this chapter

The pin interrupt generator and the pattern match engine are available on all LPC800 parts.

8.2 Features

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

8.3 Basic configuration

- Pin interrupts:
  - Select up to eight external interrupt pins from all GPIO port pins in the SYSCON block (Table 32). The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive.
  - Enable the clock to the pin interrupt register block in the SYSAHBCLKCTRL register (Table 18, bit 6).
  - If you want to use the pin interrupts to wake up the part from deep-sleep mode or power-down mode, enable the pin interrupt wake-up feature in the STARTERP0 register (Table 33).
  - Each selected pin interrupt is assigned to one interrupt in the NVIC (interrupts #24 to #31 for pin interrupts 0 to 7).
- Pattern match engine:
  - Select up to eight external pins from all GPIO port pins in the SYSCON block (Table 32). The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive.

- Enable the clock to the pin interrupt register block in the SYSAHBCLKCTRL register (Table 18, bit 6).
- Each bit slice of the pattern match engine is assigned to one interrupt in the NVIC (interrupts #24 to #31 for slices 0 to 7).
- The combined interrupt from all slices or slice combinations can be connected to the ARM RXEV request and to pin function GPIO_INT_BMAT through the switch matrix movable function register (PINASSIGN8, Table 105).

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

Follow these steps to configure pins as pin interrupts:

1. Determine the pins that serve as pin interrupts on the LPC800 package. See the data sheet for determining the GPIO port pin number associated with the package pin.
2. For each pin interrupt, program the GPIO port pin number into one of the eight PINTSEL registers in the SYSCON block.

   Remark: The port pin number serves to identify the pin to the PINTSEL register. Any function, including GPIO, can be assigned to this pin through the switch matrix.
3. Enable each pin interrupt in the NVIC.

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

See Section 4.6.27 “Pin interrupt select registers” in the SYSCON block for the PINTSEL registers.

8.4 Pin description

The inputs to the pin interrupt and pattern match engine are determined by the pin interrupt select registers in the SYSCON block. See Section 8.3.1.

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

See Section 9.3.1 “Connect an internal signal to a package pin” for the steps that you need to follow to assign the GPIO pattern match function to a pin on the LPC800 package.

Table 79. Pin interrupt/pattern match engine pin description

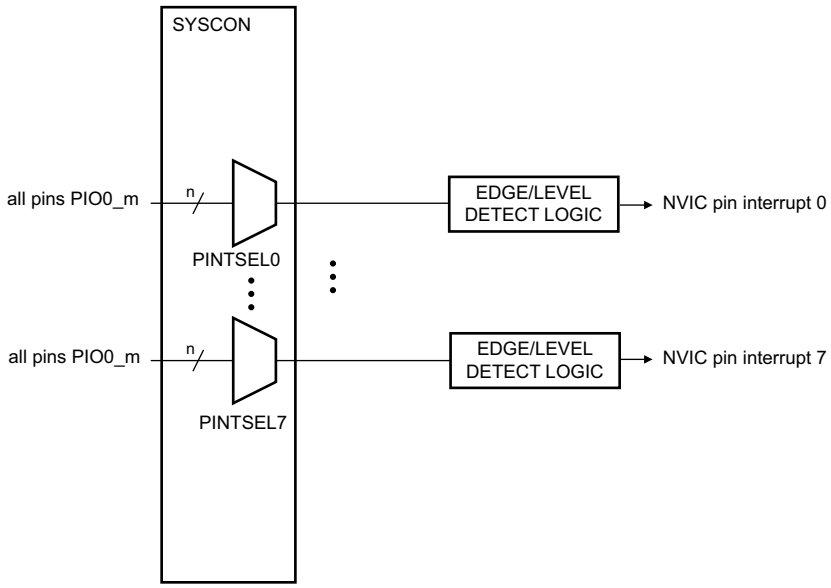
Function	Direction	Pin	Description	SWM register	Reference
GPIO_INT_BMAT	O	any	GPIO pattern match output	PINASSIGN8	Table 105

8.5 General description

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

8.5.1 Pin interrupts

From all available GPIO pins, up to eight pins can be selected in the system control block to serve as external interrupt pins (see Table 32). The external interrupt pins are connected to eight individual interrupts in the NVIC and are created based on rising or falling edges or on the input level on the pin.



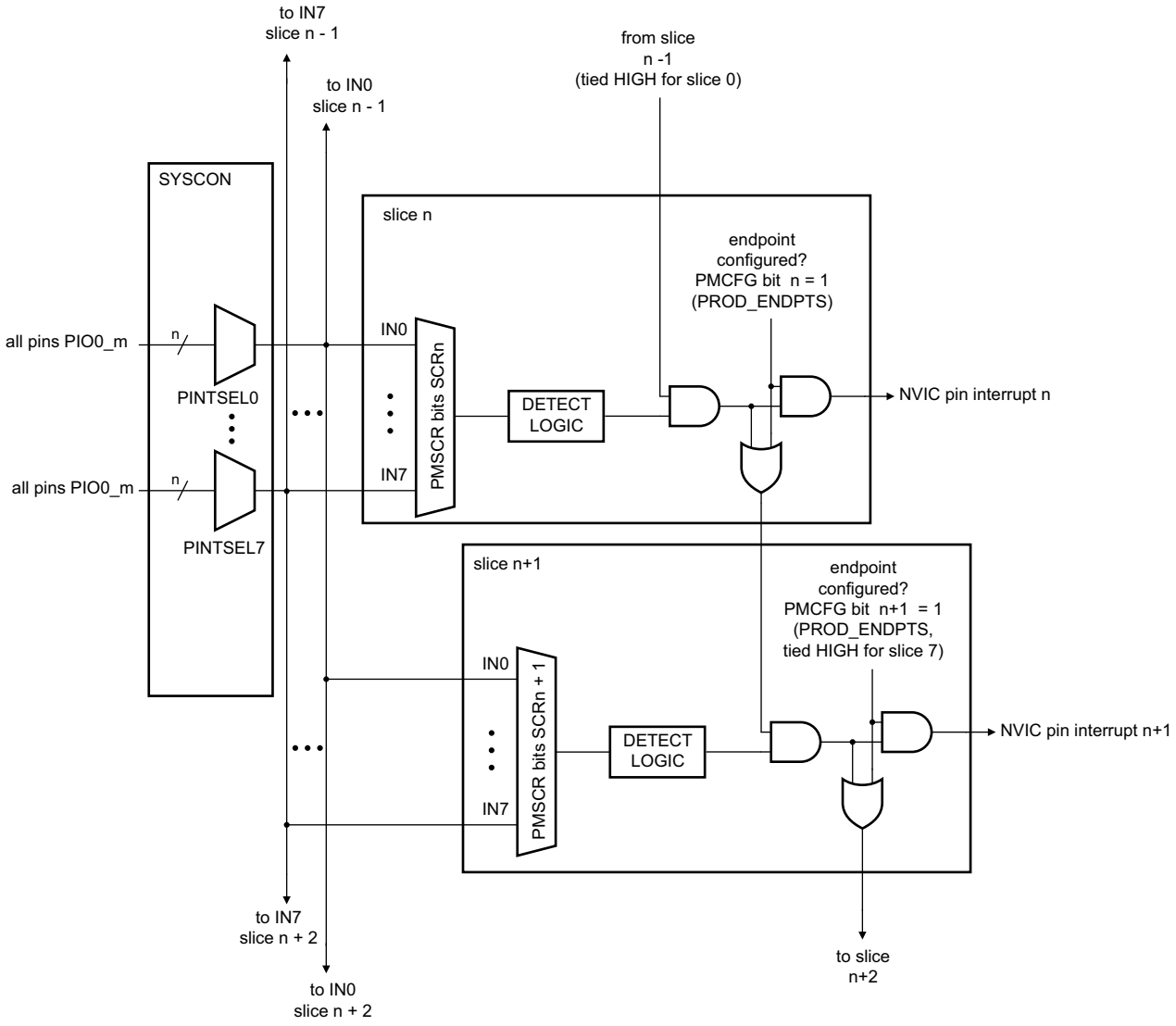
n = 6 for the DIP8 package, n= 14 for the TSSOP16 package, n = 18 for the TSSOP/SOP20 packages.

Fig 6. Pin interrupt connections

8.5.2 Pattern match engine

The pattern match feature allows complex boolean expressions to be constructed from the same set of eight GPIO pins that were selected for the GPIO pin interrupts. Each term in the boolean expression is implemented as one slice of the pattern match engine. A slice consists of an input selector and a detect logic. The slice input selector selects one input from the available eight inputs with each input connected to a pin by the input's PINTSEL register.

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



See Figure 8 for the detect logic block.

Fig 7. Pattern match engine connections

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

- Edge with memory (sticky): A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detect logic output remains HIGH) until the pattern match engine detect logic is cleared again.
- Event (non-sticky): Every time an edge (rising or falling) is detected, the detect logic output for this pin goes HIGH. This bit is cleared after one clock cycle, and the detect logic can detect another edge,
- Level: A HIGH or LOW level on the selected input.

Figure 8 shows the details of the edge detection logic for each slice.

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

You can create a time window during which rising or falling edges can create a pin interrupt by combining a level detect with an event detect. See Section 8.7.3 for details.

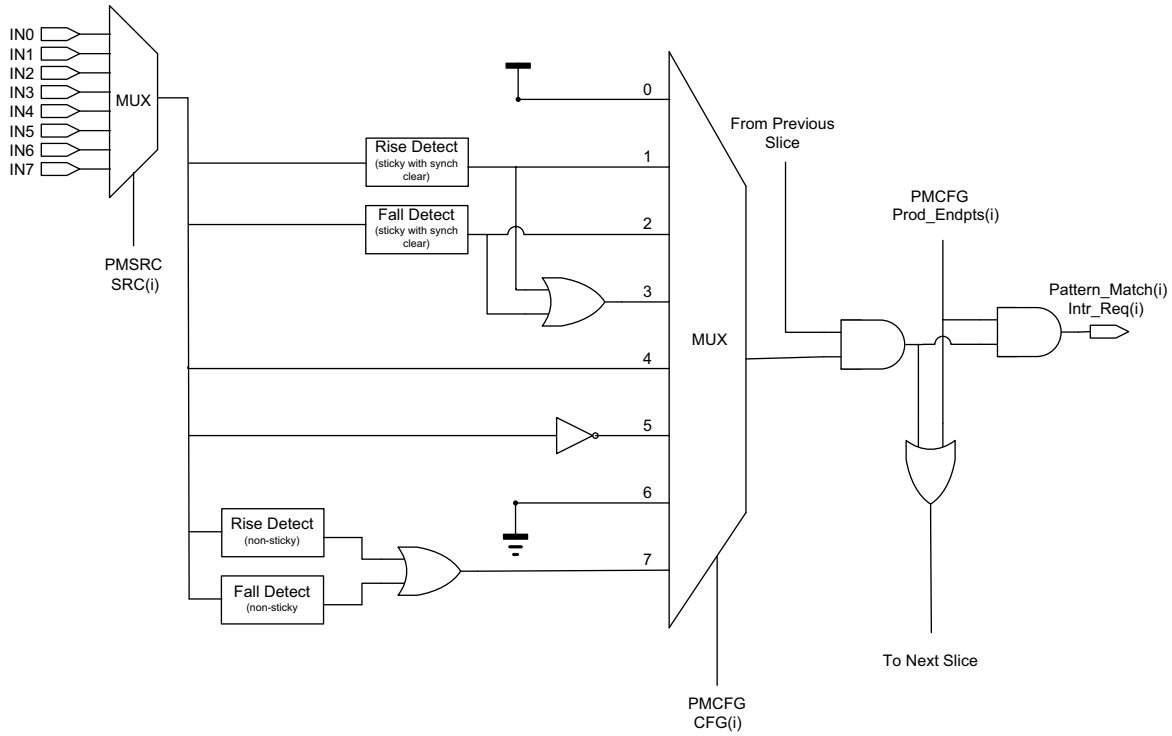


Fig 8. Pattern match bit slice with detect logic

8.5.2.1 Inputs and outputs of the pattern match engine

The connections between the pins and the pattern match engine are shown in Figure 7. All inputs to the pattern match engine are selected in the SYSCON block and can be GPIO port pins or another pin function depending on the switch matrix configuration.

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

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

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

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

Remark: Pattern matching cannot be used to wake the part up from Deep-sleep or power-down mode. Pin interrupts must be selected in order to use the pins for wake-up.

8.5.2.2 Boolean expressions

The pattern match module is constructed of eight bit-slice elements. Each bit slice is programmed to represent one component of one minterm (product term) within the boolean expression. The interrupt request associated with the last bit slice for a particular minterm will be asserted whenever that minterm is matched.
(See bit slice drawing Figure 8).

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

Example:

Assume the expression: (IN0)~(IN1)(IN3)^ + (IN1)(IN2) + (IN0)~(IN3)~(IN4) is specified through the registers PMSRC (Table 92) and PMCFG (Table 93). Each term in the boolean expression, (IN0), ~(IN1), (IN3)^, etc., represents one bit slice of the pattern match engine.

- In the first minterm (IN0)~(IN1)(IN3)^, bit slice 0 monitors for a high-level on input (IN0), bit slice 1 monitors for a low level on input (IN1) and bit slice 2 monitors for a rising-edge on input (IN3). If this combination is detected, that is if all three terms are true, the interrupt associated with bit slice 2 (PININT2_IRQ) will be asserted.
- In the second minterm (IN1)(IN2), bit slice 3 monitors input (IN1) for a high level, bit slice 4 monitors input (IN2) for a high level. If this combination is detected, the interrupt associated with bit slice 4 (PININT4_IRQ) will be asserted.
- In the third minterm (IN0)~(IN3)~(IN4), bit slice 5 monitors input (IN0) for a high level, bit slice 6 monitors input (IN3) for a low level, and bit slice 7 monitors input (IN4) for a low level. If this combination is detected, the interrupt associated with bit slice 7(PININT7_IRQ) will be asserted.

- The ORed result of all three minterms asserts the RXEV request to the CPU and the GPIO_INT_BMAT output. That is, if any of the three minterms are true, the output is asserted.

Related links:

Section 8.7.2

8.6 Register description

Table 80. Register overview: Pin interrupts and pattern match engine (base address: 0xA000 4000)

Name	Access	Address offset	Description	Reset value	Reference
ISEL	R/W	0x000	Pin Interrupt Mode register	0	Table 81
IENR	R/W	0x004	Pin interrupt level or rising edge interrupt enable register	0	Table 82
SIENR	WO	0x008	Pin interrupt level or rising edge interrupt set register	NA	Table 83
CIENR	WO	0x00C	Pin interrupt level (rising edge interrupt) clear register	NA	Table 84
IENF	R/W	0x010	Pin interrupt active level or falling edge interrupt enable register	0	Table 85
SIENF	WO	0x014	Pin interrupt active level or falling edge interrupt set register	NA	Table 86
CIENF	WO	0x018	Pin interrupt active level or falling edge interrupt clear register	NA	Table 87
RISE	R/W	0x01C	Pin interrupt rising edge register	0	Table 88
FALL	R/W	0x020	Pin interrupt falling edge register	0	Table 89
IST	R/W	0x024	Pin interrupt status register	0	Table 90
PMCTRL	R/W	0x028	Pattern match interrupt control register	0	Table 91
PMSRC	R/W	0x02C	Pattern match interrupt bit-slice source register	0	Table 92
PMCFG	R/W	0x030	Pattern match interrupt bit slice configuration register	0	Table 93

8.6.1 Pin interrupt mode register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 4.6.27), one bit in the ISEL register determines whether the interrupt is edge or level sensitive.

Table 81. Pin interrupt mode register (ISEL, address 0xA000 4000) bit description

Bit	Symbol	Description	Reset value	Access
7:0	PMODE	Selects the interrupt mode for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn. 0 = Edge sensitive 1 = Level sensitive	0	R/W
31:8	-	Reserved.	-	-

8.6.2 Pin interrupt level or rising edge interrupt enable register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 4.6.27), one bit in the IENR register enables the interrupt depending on the pin interrupt mode configured in the ISEL register:

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

Table 82. Pin interrupt level or rising edge interrupt enable register (IENR, address 0xA000 4004) bit description

Bit	Symbol	Description	Reset value	Access
7:0	ENRL	Enables the rising edge or level interrupt for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn. 0 = Disable rising edge or level interrupt. 1 = Enable rising edge or level interrupt.	0	R/W
31:8	-	Reserved.	-	-

8.6.3 Pin interrupt level or rising edge interrupt set register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 4.6.27), one bit in the SIENR register sets the corresponding bit in the IENR register depending on the pin interrupt mode configured in the ISEL register:

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

Table 83. Pin interrupt level or rising edge interrupt set register (SIENR, address 0xA000 4008) bit description

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

8.6.4 Pin interrupt level or rising edge interrupt clear register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 4.6.27), one bit in the CIENR register clears the corresponding bit in the IENR register depending on the pin interrupt mode configured in the ISEL register:

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

Table 84. Pin interrupt level or rising edge interrupt clear register (CIENR, address 0xA000 400C) bit description

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

8.6.5 Pin interrupt active level or falling edge interrupt enable register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 4.6.27), one bit in the IENF register enables the falling edge interrupt or the configures the level sensitivity depending on the pin interrupt mode configured in the ISEL register:

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

Table 85. Pin interrupt active level or falling edge interrupt enable register (IENF, address 0xA000 4010) bit description

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

8.6.6 Pin interrupt active level or falling edge interrupt set register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 4.6.27), one bit in the SIENF register sets the corresponding bit in the IENF register depending on the pin interrupt mode configured in the ISEL register:

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

Table 86. Pin interrupt active level or falling edge interrupt set register (SIENF, address 0xA000 4014) bit description

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

8.6.7 Pin interrupt active level or falling edge interrupt clear register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 4.6.27), one bit in the CIENF register sets the corresponding bit in the IENF register depending on the pin interrupt mode configured in the ISEL register:

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

Table 87. Pin interrupt active level or falling edge interrupt clear register (CIENF, address 0xA000 4018) bit description

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

8.6.8 Pin interrupt rising edge register

This register contains ones for pin interrupts selected in the PINTSELn registers (see Section 4.6.27) on which a rising edge has been detected. Writing ones to this register clears rising edge detection. Ones in this register assert an interrupt request for pins that are enabled for rising-edge interrupts. All edges are detected for all pins selected by the PINTSELn registers, regardless of whether they are interrupt-enabled.

Table 88. Pin interrupt rising edge register (RISE, address 0xA000 401C) bit description

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

8.6.9 Pin interrupt falling edge register

This register contains ones for pin interrupts selected in the PINTSELn registers (see Section 4.6.27) on which a falling edge has been detected. Writing ones to this register clears falling edge detection. Ones in this register assert an interrupt request for pins that are enabled for falling-edge interrupts. All edges are detected for all pins selected by the PINTSELn registers, regardless of whether they are interrupt-enabled.

Table 89. Pin interrupt falling edge register (FALL, address 0xA000 4020) bit description

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

8.6.10 Pin interrupt status register

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

Table 90. Pin interrupt status register (IST, address 0xA000 4024) bit description

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

8.6.11 Pattern Match Interrupt Control Register

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

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

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

Table 91. Pattern match interrupt control register (PMCTRL, address 0xA000 4028) bit description

Bit	Symbol	Value	Description	Reset value
0	SEL_PMATCH		Specifies whether the 8 pin interrupts are controlled by the pin interrupt function or by the pattern match function.	0
		0	Pin interrupt. Interrupts are driven in response to the standard pin interrupt function	
		1	Pattern match. Interrupts are driven in response to pattern matches.	
1	ENA_RXEV		Enables the RXEV output to the ARM cpu and/or to a GPIO output when the specified boolean expression evaluates to true.	0
		0	Disabled. RXEV output to the cpu is disabled.	
		1	Enabled. RXEV output to the cpu is enabled.	
23:2	-		Reserved. Do not write 1s to unused bits.	0
31:24	PMAT	-	This field displays the current state of pattern matches. A 1 in any bit of this field indicates that the corresponding product term is matched by the current state of the appropriate inputs.	0x0

8.6.12 Pattern Match Interrupt Bit-Slice Source register

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

Each of the possible eight inputs is selected in the pin interrupt select registers in the SYSCON block. See Section 4.6.27. Input 0 corresponds to the pin selected in the PINTSEL0 register, input 1 corresponds to the pin selected in the PINTSEL1 register, and so forth.

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

Table 92. Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description

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

Table 92. Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description

Bit	Symbol	Value	Description	Reset value
10:8	SRC0		Selects the input source for bit slice 0	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 0.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 0.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 0.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 0.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 0.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 0.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 0.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 0.	
13:11	SRC1		Selects the input source for bit slice 1	0
		0x0	Input 0. Selects pin interrupt input 0 as the source to bit slice 1.	
		0x1	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 0.	
		0x2	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 0.	
		0x3	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 0.	
		0x4	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 0.	
		0x5	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 0.	
		0x6	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 0.	
		0x7	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 0.	

Table 92. Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description

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

Table 92. Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description

Bit	Symbol	Value	Description	Reset value
22:20	SRC4		Selects the input source for bit slice 4	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 0.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 0.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 0.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 0.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 0.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 0.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 0.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 0.	
25:23	SRC5		Selects the input source for bit slice 5	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 0.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 0.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 0.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 0.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 0.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 0.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 0.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 0.	

Table 92. Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description

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

8.6.13 Pattern Match Interrupt Bit Slice Configuration register

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

Two types of edge detection on each input are possible:

- Sticky: A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detect logic output remains HIGH) until the pattern match engine detect logic is cleared again.
- Non-sticky: Every time an edge (rising or falling) is detected, the detect logic output for this pin goes HIGH. This bit is cleared after one clock cycle, and the edge detect logic can detect another edge,

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

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

1. The interrupt request associated with this bit slice will be asserted whenever a match to that product term is detected.
2. The next bit slice will start a new, independent product term in the boolean expression (i.e. an OR will be inserted in the boolean expression following the element controlled by this bit slice).

Table 93. Pattern match bit slice configuration register (PMCFG, address 0xA000 4030) bit description

Bit	Symbol	Value	Description	Reset value
0	PROD_ENDPTS0		Determines whether slice 0 is an endpoint.	0
		0	No effect. Slice 0 is not an endpoint.	
		1	endpoint. Slice 0 is the endpoint of a product term (minterm). Pin interrupt 0 in the NVIC is raised if the minterm evaluates as true.	
1	PROD_ENDPTS1		Determines whether slice 1 is an endpoint.	0
		0	No effect. Slice 1 is not an endpoint.	
		1	endpoint. Slice 1 is the endpoint of a product term (minterm). Pin interrupt 1 in the NVIC is raised if the minterm evaluates as true.	
2	PROD_ENDPTS2		Determines whether slice 2 is an endpoint.	0
		0	No effect. Slice 2 is not an endpoint.	
		1	endpoint. Slice 2 is the endpoint of a product term (minterm). Pin interrupt 2 in the NVIC is raised if the minterm evaluates as true.	
3	PROD_ENDPTS3		Determines whether slice 3 is an endpoint.	0
		0	No effect. Slice 3 is not an endpoint.	
		1	endpoint. Slice 3 is the endpoint of a product term (minterm). Pin interrupt 3 in the NVIC is raised if the minterm evaluates as true.	
4	PROD_ENDPTS4		Determines whether slice 4 is an endpoint.	0
		0	No effect. Slice 4 is not an endpoint.	
		1	endpoint. Slice 4 is the endpoint of a product term (minterm). Pin interrupt 4 in the NVIC is raised if the minterm evaluates as true.	
5	PROD_ENDPTS5		Determines whether slice 5 is an endpoint.	0
		0	No effect. Slice 5 is not an endpoint.	
		1	endpoint. Slice 5 is the endpoint of a product term (minterm). Pin interrupt 5 in the NVIC is raised if the minterm evaluates as true.	

Table 93. Pattern match bit slice configuration register (PMCFG, address 0xA000 4030) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
6	PROD_ENDPTS6		Determines whether slice 6 is an endpoint.	0
		0	No effect. Slice 6 is not an endpoint.	
		1	endpoint. Slice 6 is the endpoint of a product term (minterm). Pin interrupt 6 in the NVIC is raised if the minterm evaluates as true.	
7	-		Reserved. Bit slice 7 is automatically considered a product end point.	0
10:8	CFG0		Specifies the match contribution condition for bit slice 0.	0b000
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
13:11	CFG1		Specifies the match contribution condition for bit slice 1.	0b000
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

Table 93. Pattern match bit slice configuration register (PMCFG, address 0xA000 4030) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
16:14	CFG2		Specifies the match contribution condition for bit slice 2.	0b000
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
19:17	CFG3		Specifies the match contribution condition for bit slice 3.	0b000
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

Table 93. Pattern match bit slice configuration register (PMCFG, address 0xA000 4030) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
22:20	CFG4		Specifies the match contribution condition for bit slice 4.	0b000
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
25:23	CFG5		Specifies the match contribution condition for bit slice 5.	0b000
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

Table 93. Pattern match bit slice configuration register (PMCFG, address 0xA000 4030) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
28:26	CFG6		Specifies the match contribution condition for bit slice 6.	0b000
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
31:29	CFG7		Specifies the match contribution condition for bit slice 7.	0b000
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

8.7 Functional description

8.7.1 Pin interrupts

In this interrupt facility, up to 8 pins are identified as interrupt sources by the Pin Interrupt Select registers (PINTSEL0-7). All registers in the pin interrupt block contain 8 bits, corresponding to the pins called out by the PINTSEL0-7 registers. The ISEL register defines whether each interrupt pin is edge- or level-sensitive. The RISE and FALL registers detect edges on each interrupt pin, and can be written to clear (and set) edge detection. The IST register indicates whether each interrupt pin is currently requesting an interrupt, and this register can also be written to clear interrupts.

The other pin interrupt registers play different roles for edge-sensitive and level-sensitive pins, as described in Table 94.

Table 94. Pin interrupt registers for edge- and level-sensitive pins

Name	Edge-sensitive function	Level-sensitive function
IENR	Enables rising-edge interrupts.	Enables level interrupts.
SIENR	Write to enable rising-edge interrupts.	Write to enable level interrupts.
CIENR	Write to disable rising-edge interrupts.	Write to disable level interrupts.
IENF	Enables falling-edge interrupts.	Selects active level.
SIENF	Write to enable falling-edge interrupts.	Write to select high-active.
CIENF	Write to disable falling-edge interrupts.	Write to select low-active.

8.7.2 Pattern Match engine example

Suppose the desired boolean pattern to be matched is:
(IN1) + (IN1 * IN2) + (~IN2 * ~IN3 * IN6fe) + (IN5 * IN7ev)

with:

IN6fe = (sticky) falling-edge on input 6

IN7ev = (non-sticky) event (rising or falling edge) on input 7

Each individual term in the expression shown above is controlled by one bit-slice. To specify this expression, program the pattern match bit slice source and configuration register fields as follows:

- PMSRC register (Table 92):
  - Since bit slice 5 will be used to detect a sticky event on input 6, you can write a 1 to the SRC5 bits to clear any pre-existing edge detects on bit slice 5.
  - SRC0: 001 - select input 1 for bit slice 0
  - SRC1: 001 - select input 1 for bit slice 1
  - SRC2: 010 - select input 2 for bit slice 2
  - SRC3: 010 - select input 2 for bit slice 3
  - SRC4: 011 - select input 3 for bit slice 4
  - SRC5: 110 - select input 6 for bit slice 5
  - SRC6: 101 - select input 5 for bit slice 6

- SRC7: 111 - select input 7 for bit slice 7

- PMCFG register (Table 93):
  - PROD_ENDPTS0 = 1
  - PROD_ENDPTS02 = 1
  - PROD_ENDPTS5 = 1
  - All other slices are not product term endpoints and their PROD_ENDPTS bits are 0. Slice 7 is always a product term endpoint and does not have a register bit associated with it.
  - = 0100101 - bit slices 0, 2, 5, and 7 are product-term endpoints. (Bit slice 7 is an endpoint by default - no associated register bit).
  - CFG0: 000 - high level on the selected input (input 1) for bit slice 0
  - CFG1: 000 - high level on the selected input (input 1) for bit slice 1
  - CFG2: 000 - high level on the selected input (input 2) for bit slice 2
  - CFG3: 101 - low level on the selected input (input 2) for bit slice 3
  - CFG4: 101 - low level on the selected input (input 3) for bit slice 4
  - CFG5: 010 - (sticky) falling edge on the selected input (input 6) for bit slice 5
  - CFG6: 000 - high level on the selected input (input 5) for bit slice 6
  - CFG7: 111 - event (any edge, non-sticky) on the selected input (input 7) for bit slice 7
- PMCTRL register (Table 91):
  - Bit0: Setting this bit will select pattern matches to generate the pin interrupts in place of the normal pin interrupt mechanism.

    For this example, pin interrupt 0 will be asserted when a match is detected on the first product term (which, in this case, is just a high level on input 1).

    Pin interrupt 2 will be asserted in response to a match on the second product term.

    Pin interrupt 5 will be asserted when there is a match on the third product term.

    Pin interrupt 7 will be asserted on a match on the last term.
  - Bit1: Setting this bit will cause the RxEv signal to the ARM CPU to be asserted whenever a match occurs on ANY of the product terms in the expression. Otherwise, the RXEV line will not be used.
  - Bit31:24: At any given time, bits 0, 2, 5 and/or 7 may be high if the corresponding product terms are currently matching.
  - The remaining bits will always be low.

8.7.3 Pattern match engine edge detect examples

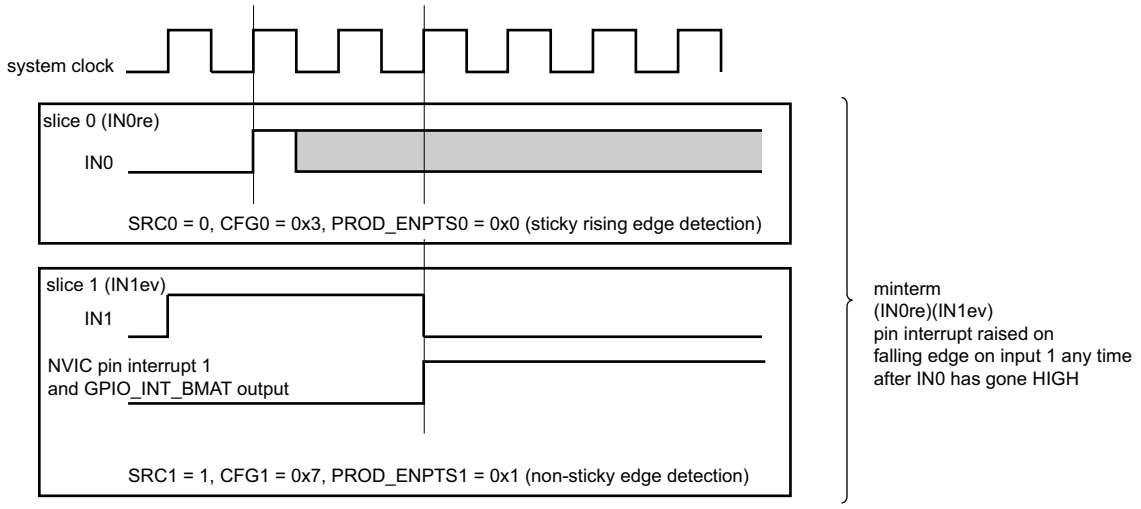


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 9. Pattern match engine examples: sticky edge detect

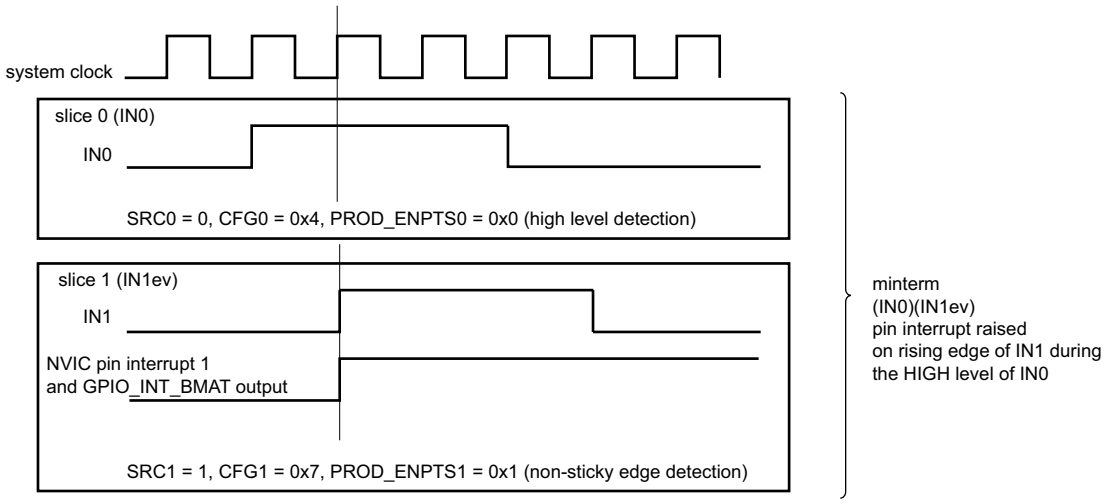


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 10. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true

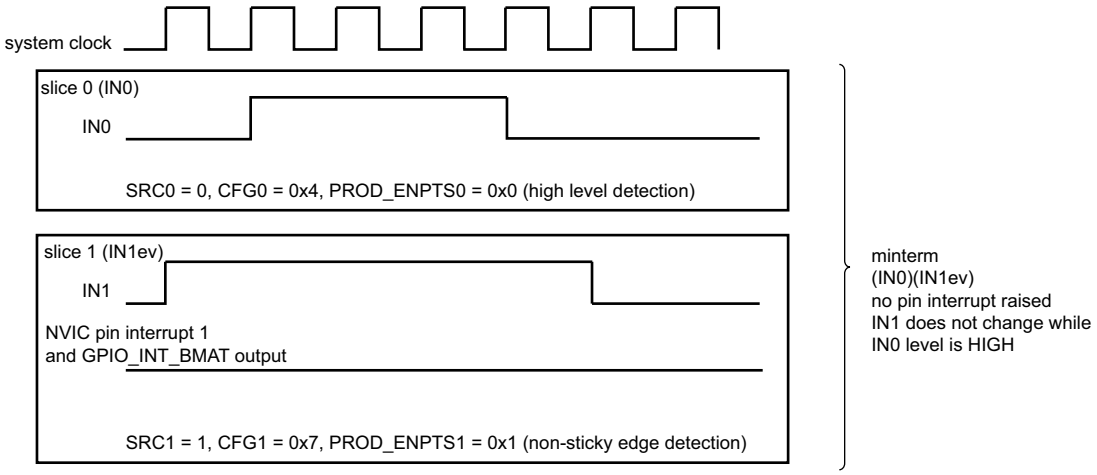


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 11. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false

9.1 How to read this chapter

The switch matrix is identical for all LPC800 parts. The USART2 and SPI1 functions are only available on parts LPC812M101FDH20 and LPC812M101FDH16 and the corresponding switch matrix select bits are reserved for all other parts.

9.2 Features

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

9.3 Basic configuration

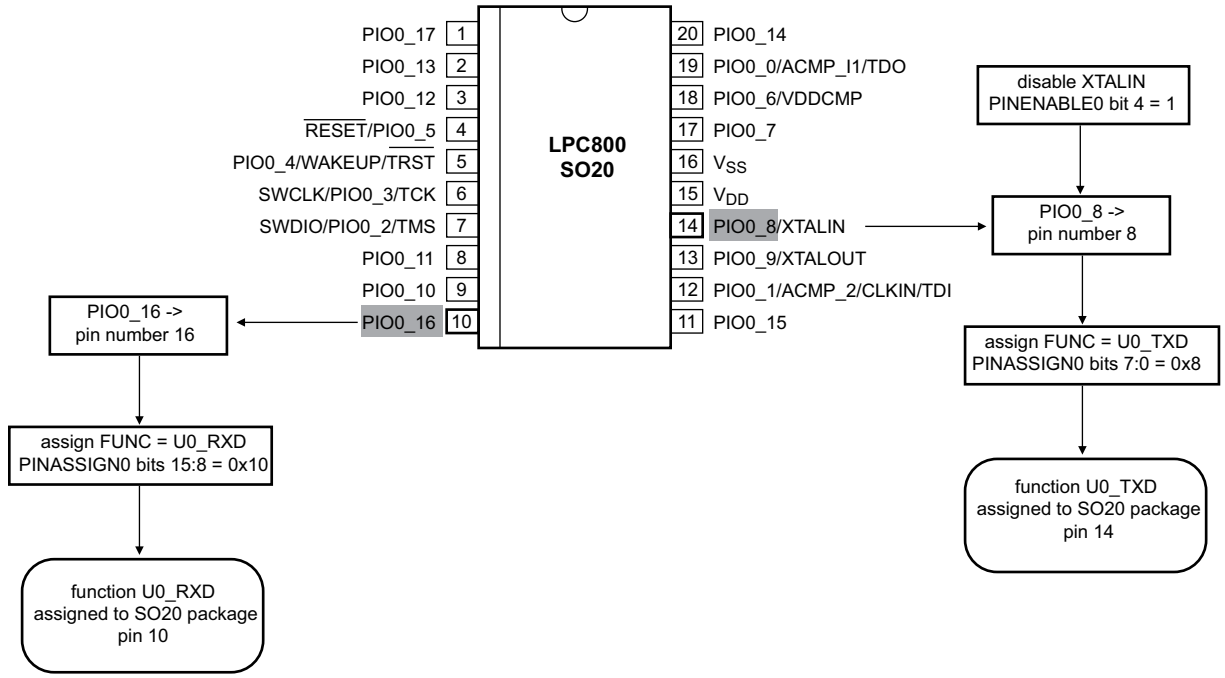
Once configured, no clocks are needed for the switch matrix to function. The system clock is needed only to write to or read from the pin assignment registers. After the switch matrix is configured, disable the clock to the switch matrix block in the SYSAHBCLKCTRL register.

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

The boot loader assigns the SWD functions to pins PIO0_2 and PIO0_3. If the user code disables the SWD functions through the switch matrix to use the pins for other functions, the SWD port is disabled.

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

9.3.1 Connect an internal signal to a package pin



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

Fig 12. Example: Connect function U0_RXD and U0_TXD to pins 10 and 14 on the SO20 package

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

1. Find the function FUNC in the list of movable functions in Table 95 or in the data sheet.
2. Use the LPC800 data sheet to decide which pin x on the LPC800 package to connect FUNC to.
3. Use the pin description table to find the default GPIO function PIO0_n assigned to package pin x. m is the pin number.
4. Locate the pin assignment register for the function FUNC in the switch matrix register description.
5. Disable any special functions on pin PIO0_n in the PINENABLE0 register.
6. Program the pin number n into the bits assigned to FUNC.

FUNC is now connected to pin x on the package.

9.3.2 Enable an analog input or other special function

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

- If you want to assign a GPIO pin to a pin on any LPC800 package, disable any special function available on this pin in the PINENABLE0 register and do not assign any movable function to it.

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

9.4 General description

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

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

On the LPC800, most functions can be assigned through the switch matrix to any external pin that is not a power or ground pin. These functions are called movable functions.

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

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

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

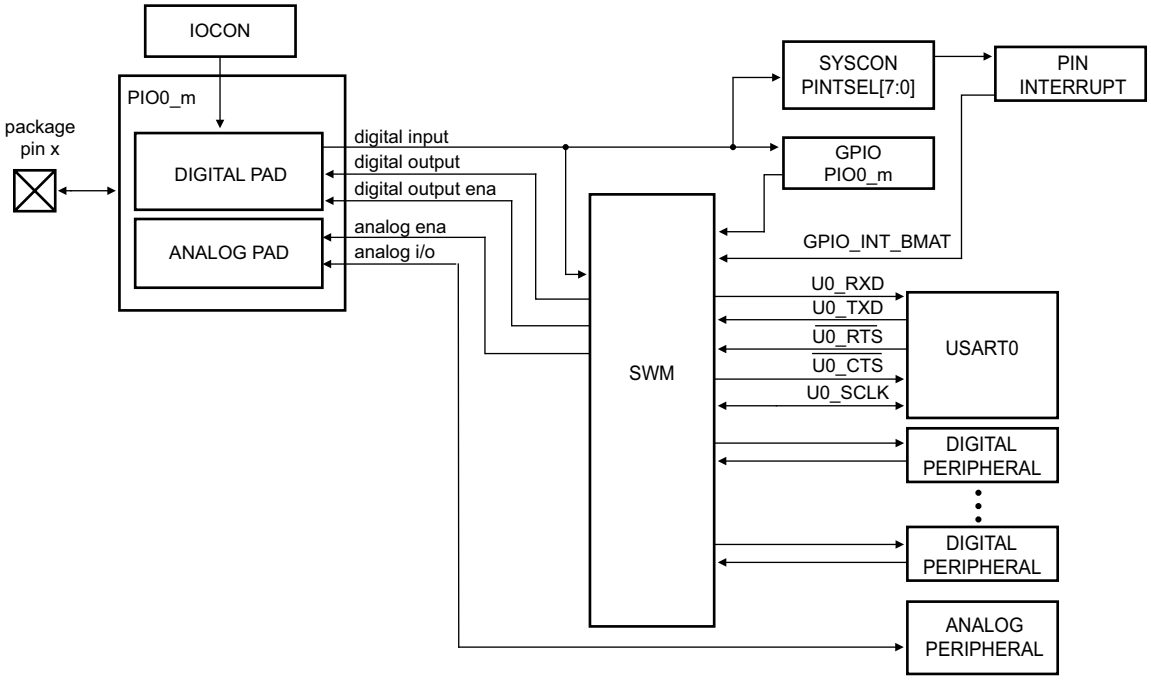


Fig 13. Functional diagram of the switch matrix

Remark: From all available movable and fixed-pin functions, you can assign multiple functions to the same pin but no more than one output or bidirectional function (see Figure 13). Use the following guidelines when assigning pins:

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

  Example:

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

  You can send the input on one pin to all SCT inputs to use as an SCT abort signal.
- It is allowed to let one digital output function control one or more digital inputs by programming the same pin number in the PINASSIGN register bit fields for the output and inputs.

  Example:

  You can assign the same pin number to the ACMP_OUT function and an SCT input CTIN_n. This connects the comparator output to input n of the SCT.

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

9.4.1 Movable functions

Table 95. Movable functions (assign to pins PIO0_0 to PIO0_17 through switch matrix)

Function name	Type	Description	SWM Pin assign register	Reference
U0_TXD	O	Transmitter output for USART0.	PINASSIGN0	Table 97
U0_RXD	I	Receiver input for USART0.	PINASSIGN0	Table 97
$\overline{\text{U0\_RTS}}$	O	Request To Send output for USART0.	PINASSIGN0	Table 97
$\overline{\text{U0\_CTS}}$	I	Clear To Send input for USART0.	PINASSIGN0	Table 97
U0_SCLK	I/O	Serial clock input/output for USART0 in synchronous mode.	PINASSIGN1	Table 98
U1_TXD	O	Transmitter output for USART1.	PINASSIGN1	Table 98
U1_RXD	I	Receiver input for USART1.	PINASSIGN1	Table 98
$\overline{\text{U1\_RTS}}$	O	Request To Send output for USART1.	PINASSIGN1	Table 98
$\overline{\text{U1\_CTS}}$	I	Clear To Send input for USART1.	PINASSIGN2	Table 99
U1_SCLK	I/O	Serial clock input/output for USART1 in synchronous mode.	PINASSIGN2	Table 99
U2_TXD	O	Transmitter output for USART2.	PINASSIGN2	Table 99
U2_RXD	I	Receiver input for USART2.	PINASSIGN2	Table 99
$\overline{\text{U2\_RTS}}$	O	Request To Send output for USART1.	PINASSIGN3	Table 100
$\overline{\text{U2\_CTS}}$	I	Clear To Send input for USART1.	PINASSIGN3	Table 100
U2_SCLK	I/O	Serial clock input/output for USART1 in synchronous mode.	PINASSIGN3	Table 100
SPI0_SCK	I/O	Serial clock for SPI0.	PINASSIGN3	Table 100
SPI0_MOSI	I/O	Master Out Slave In for SPI0.	PINASSIGN4	Table 101
SPI0_MISO	I/O	Master In Slave Out for SPI0.	PINASSIGN4	Table 101
SPI0_SSEL	I/O	Slave select for SPI0.	PINASSIGN4	Table 101
SPI1_SCK	I/O	Serial clock for SPI1.	PINASSIGN4	Table 101
SPI1_MOSI	I/O	Master Out Slave In for SPI1.	PINASSIGN5	Table 102
SPI1_MISO	I/O	Master In Slave Out for SPI1.	PINASSIGN5	Table 102
SPI1_SSEL	I/O	Slave select for SPI1.	PINASSIGN5	Table 102
CTIN_0	I	SCT input 0.	PINASSIGN5	Table 102
CTIN_1	I	SCT input 1.	PINASSIGN6	Table 103
CTIN_2	I	SCT input 2.	PINASSIGN6	Table 103
CTIN_3	I	SCT input 3.	PINASSIGN6	Table 103
CTOUT_0	O	SCT output 0.	PINASSIGN6	Table 103
CTOUT_1	O	SCT output 1.	PINASSIGN7	Table 104
CTOUT_2	O	SCT output 2.	PINASSIGN7	Table 104
CTOUT_3	O	SCT output 3.	PINASSIGN7	Table 104
I2C0_SDA	I/O	I^2C-bus data input/output (open-drain if assigned to pin PIO0_11). High-current sink only if assigned to pin PIO0_11 and if I^2C Fast-mode Plus is selected in the I/O configuration register.	PINASSIGN7	Table 104

Table 95. Movable functions (assign to pins PIO0_0 to PIO0_17 through switch matrix)

Function name	Type	Description	SWM Pin assign register	Reference
I2C0_SCL	I/O	I^2C-bus clock input/output (open-drain if assigned to pin PIO0_10). High-current sink only if assigned to PIO0_10 and if I^2C Fast-mode Plus is selected in the I/O configuration register.	PINASSIGN8	Table 105
ACMP_O	O	Analog comparator output.	PINASSIGN8	Table 105
CLKOUT	O	Clock output.	PINASSIGN8	Table 105
GPIO_INT_BMAT	O	Output of the pattern match engine.	PINASSIGN8	Table 105

9.4.2 Switch matrix register interface

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

GPIOs range from PIO0_0 to PIO0_17 and, for assignment through the pin-assignment registers, are numbered 0 to 17.

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

1. **Movable functions** (PINASSIGN0 to 8):

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

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

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

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

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

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

   - If a fixed-pin function is deselected, any movable function can be assigned to its port and pin.
   - If a fixed-pin function is deselected and no movable function is assigned to this pin, the pin is assigned its GPIO function.
   - On reset, all fixed-pin functions are deselected.
   - If a fixed-pin analog function is selected, its assigned pin cannot be used for any other function.

9.5 Register description

Table 96. Register overview: Switch matrix (base address 0x4000 C000)

Name	Access	Offset	Description	Reset value	Reference
PINASSIGN0	R/W	0x000	Pin assign register 0. Assign movable functions U0_TXD, U0_RXD, U0_RTS, U0_CTS.	0xFFFF FFFF	Table 97
PINASSIGN1	R/W	0x004	Pin assign register 1. Assign movable functions U0_SCLK, U1_TXD, U1_RXD, U1_RTS.	0xFFFF FFFF	Table 98
PINASSIGN2	R/W	0x008	Pin assign register 2. Assign movable functions U1_CTS, U1_SCLK, U2_TXD, U2_RXD.	0xFFFF FFFF	Table 99
PINASSIGN3	R/W	0x00C	Pin assign register 3. Assign movable function U2_RTS, U2_CTS, U2_SCLK, SPI0_SCK.	0xFFFF FFFF	Table 100
PINASSIGN4	R/W	0x010	Pin assign register 4. Assign movable functions SPI0_MOSI, SPI0_MISO, SPI0_SSEL, SPI1_SCK.	0xFFFF FFFF	Table 101
PINASSIGN5	R/W	0x014	Pin assign register 5. Assign movable functions SPI1_MOSI, SPI1_MISO, SPI1_SSEL, CTIN_0	0xFFFF FFFF	Table 102
PINASSIGN6	R/W	0x018	Pin assign register 6. Assign movable functions CTIN_1, CTIN_2, CTIN_3, CTOUT_0.	0xFFFF FFFF	Table 103
PINASSIGN7	R/W	0x01C	Pin assign register 7. Assign movable functions CTOUT_1, CTOUT_2, CTOUT_3, I2C_SDA.	0xFFFF FFFF	Table 104
PINASSIGN8	R/W	0x020	Pin assign register 8. Assign movable functions I2C_SCL, ACMP_O, CLKOUT, GPIO_INT_BMAT.	0xFFFF FFFF	Table 105
-	-	0x024	Reserved.	-	-
PINENABLE0	R/W	0x1C0	Pin enable register 0. Enables fixed-pin functions ACMP_I0, ACMP_I1, SWCLK, SWDIO, XTALIN, XTALOUT, RESET, CLKIN, VDDCMP.	0x1B3	Table 106

9.5.1 Pin assign register 0

Table 97. Pin assign register 0 (PINASSIGN0, address 0x4000 C000) bit description

Bit	Symbol	Description	Reset value
7:0	U0_TXD_O	U0_TXD function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF

Table 97. Pin assign register 0 (PINASSIGN0, address 0x4000 C000) bit description

Bit	Symbol	Description	Reset value
15:8	U0_RXD_I	U0_RXD function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
23:16	U0_RTS_O	U0_RTS function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
31:24	U0_CTS_I	U0_CTS function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF

9.5.2 Pin assign register 1

Table 98. Pin assign register 1 (PINASSIGN1, address 0x4000 C004) bit description

Bit	Symbol	Description	Reset value
7:0	U0_SCLK_IO	U0_SCLK function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
15:8	U1_TXD_O	U1_TXD function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
23:16	U1_RXD_I	U1_RXD function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
31:24	U1_RTS_O	U1_RTS function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF

9.5.3 Pin assign register 2

Table 99. Pin assign register 2 (PINASSIGN2, address 0x4000 C008) bit description

Bit	Symbol	Description	Reset value
7:0	U1_CTS_I	U1_CTS function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
15:8	U1_SCLK_IO	U1_SCLK function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
23:16	U2_TXD_O	U2_TXD function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
31:24	U2_RXD_I	U2_RXD function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF

9.5.4 Pin assign register 3

Table 100. Pin assign register 3 (PINASSIGN3, address 0x4000 C00C) bit description

Bit	Symbol	Description	Reset value
7:0	U2_RTS_O	U2_RTS function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
15:8	U2_CTS_I	U2_CTS function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
23:16	U2_SCLK_IO	U2_SCLK function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
31:24	SPI0_SCK_IO	SPI0_SCK function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF

9.5.5 Pin assign register 4

Table 101. Pin assign register 4 (PINASSIGN4, address 0x4000 C010) bit description

Bit	Symbol	Description	Reset value
7:0	SPI0_MOSI_IO	SPI0_MOSI function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
15:8	SPI0_MISO_IO	SPI0_MISIO function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
23:16	SPI0_SSEL_IO	SPI0_SSEL function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
31:24	SPI1_SCK_IO	SPI1_SCK function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF

9.5.6 Pin assign register 5

Table 102. Pin assign register 5 (PINASSIGN5, address 0x4000 C014) bit description

Bit	Symbol	Description	Reset value
7:0	SPI1_MOSI_IO	SPI1_MOSI function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF

Table 102. Pin assign register 5 (PINASSIGN5, address 0x4000 C014) bit description

Bit	Symbol	Description	Reset value
15:8	SPI1_MISO_IO	SPI1_MISIO function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
23:16	SPI1_SSEL_IO	SPI1_SSEL function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
31:24	CTIN_0_I	CTIN_0 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF

9.5.7 Pin assign register 6

Table 103. Pin assign register 6 (PINASSIGN6, address 0x4000 C018) bit description

Bit	Symbol	Description	Reset value
7:0	CTIN_1_I	CTIN_1 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
15:8	CTIN_2_I	CTIN_2function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
23:16	CTIN_3_I	CTIN_3 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
31:24	CTOUT_0_O	CTOUT_0 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF

9.5.8 Pin assign register 7

Table 104. Pin assign register 7 (PINASSIGN7, address 0x4000 C01C) bit description

Bit	Symbol	Description	Reset value
7:0	CTOUT_1_O	CTOUT_1 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
15:8	CTOUT_2_O	CTOUT_2 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
23:16	CTOUT_3_O	CTOUT_3 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
31:24	I2C_SDA_IO	I2C_SDA function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF

9.5.9 Pin assign register 8

Table 105. Pin assign register 8 (PINASSIGN8, address 0x4000 C020) bit description

Bit	Symbol	Description	Reset value
7:0	I2C_SCL_IO	I2C_SCL function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
15:8	ACMP_O_O	ACMP_O_O function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
23:16	CLKOUT_O	CLKOUT function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF
31:24	GPIO_INT_BMAT_O	GPIO_INT_BMAT function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_17 (= 0x11).	0xFF

9.5.10 Pin enable register 0

Table 106. Pin enable register 0 (PINENABLE0, address 0x4000 C1C0) bit description

Bit	Symbol	Value	Description	Reset value
0	ACMP_I1_EN		Enables fixed-pin function. Writing a 1 deselects the function and any movable function can be assigned to this pin. By default the fixed--pin function is deselected and GPIO is assigned to this pin.	1
		0	Enable ACMP_I1. This function is enabled on pin PIO0_0.	
		1	Disable ACMP_I1. GPIO function PIO0_0 (default) or any other movable function can be assigned to pin PIO0_0.	
1	ACMP_I2_EN		Enables fixed-pin function. Writing a 1 deselects the function and any movable function can be assigned to this pin. By default the fixed-pin function is deselected and GPIO is assigned to this pin. Functions CLKIN and ACMP_I2 are connected to the same pin PIO0_1. To use ACMP_I2, disable the CLKIN function in bit 7 of this register and enable ACMP_I2.	1
		0	Enable ACMP_I2. This function is enabled on pin PIO0_1.	
		1	Disable ACMP_I2. GPIO function PIO0_1 (default) or any other movable function can be assigned to pin PIO0_1.	
2	SWCLK_EN		Enables fixed-pin function. Writing a 1 deselects the function and any movable function can be assigned to this pin. This function is selected by default.	0
		0	Enable SWCLK. This function is enabled on pin PIO0_3.	
		1	Disable SWCLK. GPIO function PIO0_3 is selected on this pin. Any other movable function can be assigned to pin PIO0_3.	
3	SWDIO_EN		Enables fixed-pin function. Writing a 1 deselects the function and any movable function can be assigned to this pin. This function is selected by default.	0
		0	Enable SWDIO. This function is enabled on pin PIO0_2.	
		1	Disable SWDIO. GPIO function PIO0_2 is selected on this pin. Any other movable function can be assigned to pin PIO0_2.	

Table 106. Pin enable register 0 (PINENABLE0, address 0x4000 C1C0) bit description

Bit	Symbol	Value	Description	Reset value
4	XTALIN_EN		Enables fixed-pin function. Writing a 1 deselects the function and any movable function can be assigned to this pin. By default the fixed--pin function is deselected and GPIO is assigned to this pin.	1
		0	Enable XTALIN. This function is enabled on pin PIO0_8.	
		1	Disable XTALIN. GPIO function PIO0_8 (default) or any other movable function can be assigned to pin PIO0_8.	
5	XTALOUT_EN		Enables fixed-pin function. Writing a 1 deselects the function and any movable function can be assigned to this pin. By default the fixed--pin function is deselected and GPIO is assigned to this pin.	1
		0	Enable XTALOUT. This function is enabled on pin PIO0_9.	
		1	Disable XTALOUT. GPIO function PIO0_9 (default) or any other movable function can be assigned to pin PIO0_9.	
6	RESET_EN		Enables fixed-pin function. Writing a 1 deselects the function and any movable function can be assigned to this pin. This function is selected by default.	0
		0	Enable $\overline{\text{RESET}}$. This function is enabled on pin PIO0_5.	
		1	Disable $\overline{\text{RESET}}$. GPIO function PIO0_5 is selected on this pin. Any other movable function can be assigned to pin PIO0_5.	
7	CLKIN		Enables fixed-pin function. Writing a 1 deselects the function and any movable function can be assigned to this pin. By default the fixed-pin function is deselected and GPIO is assigned to this pin. Functions CLKIN and ACMP_I2 are connected to the same pin PIO0_1. To use CLKIN, disable ACMP_I2 in bit 1 of this register and enable CLKIN.	1
		0	Enable CLKIN. This function is enabled on pin PIO0_1.	
		1	Disable CLKIN. GPIO function PIO0_1 (default) or any other movable function can be assigned to pin CLKIN.	
8	VDDCMP		Enables fixed-pin function. Writing a 1 deselects the function and any movable function can be assigned to this pin. By default the fixed--pin function is deselected and GPIO is assigned to this pin.	1
		0	Enable VDDCMP. This function is enabled on pin PIO0_6.	
		1	Disable VDDCMP. GPIO function PIO0_6 (default) or any other movable function can be assigned to pin PIO0_6.	
31:9	-		Reserved.	<tbd>

10.1 How to read this chapter

The SCT is available on all LPC800 parts.

10.2 Features

- Two 16-bit counters or one 32-bit counter.
- Counters clocked by bus clock or selected input.
- Up counters or up-down counters.
- State variable allows sequencing across multiple counter cycles.
- The following conditions define an event: a counter match condition, an input (or output) condition, a combination of a match and/or and input/output condition in a specified state, and the count direction.
- Events control outputs, interrupts, and the SCT states.
  - Match register 0 can be used as an automatic limit.
  - In bi-directional mode, events can be enabled based on the count direction.
  - Match events can be held until another qualifying event occurs.
- Selected events can limit, halt, start, or stop a counter.
- Supports:
  - 4 inputs
  - 4 outputs
  - 5 match/capture registers
  - 6 events
  - 2 states

10.3 Basic configuration

Configure the SCT as follows:

- Use the SYSAHBCLKCTRL register (Table 18) to enable the clock to the SCT register interface and peripheral clock. The LPC800 system clock is the input clock to the SCT clock processing and is the source of the SCT clock.
- Clear the SCT peripheral reset using the PRESETCTRL register (Table 7).
- The SCT combined interrupt is connected to slot #8 in the NVIC.
- Use the switch matrix to connect the SCT inputs and outputs to pins (see Section 10.4) and internally (see Section 10.5).

10.3.1 Use the SCT as a simple timer

To configure the SCT as a simple timer with match or capture functionality, follow these steps:

1. Set up the SCT as one 32-bit timer or one or two 16-bit timers. See Table 109.
2. Preload the 32-bit timer or the 16-bit timers with a count value. See Table 115.
3. If you want to create a match event when the timer reaches a match value:
   a. Configure the register map for match registers. See Table 118.
   b. Configure one or more match registers with a match value. See Table 126.
   c. For each match value, create a match event. See Table 131.
   d. If you want to create an interrupt on a match event, enable the event for interrupt. See Table 123.
   e. If you want to create a match output on a pin, connect the CTOUTn function to a pin (see Section 10.4) and select an output for the match event in the EVn_CTRL register. See Table 131. The EVn_CTRL registers also control what type of output signal is created.
4. If you want to capture a timer value on a capture signal:
   a. Configure the register map for capture registers. See Table 118.
   b. Create one or more capture events. See Table 131.
   c. Connect the CTIN functions to pins (see Section 10.4) and configure the signal to create an event. See Table 131.
5. Start the timer by writing to the CRTL register. See Table 110.
6. Read the capture registers to read the timer value at the time of the capture events.

10.4 Pin description

The SCT inputs and outputs are movable functions and are assigned to external pins through the switch matrix.

See Section 9.3.1 “Connect an internal signal to a package pin” to assign the SCT functions to pins on the LPC800 package.

Table 107. SCT pin description

Function	Direction	Pin	Description	SWM register	Reference
CTIN_0	I	any	SCT input 0	PINASSIGN5	Table 102
CTIN_1	I	any	SCT input 1	PINASSIGN6	Table 103
CTIN_2	I	any	SCT input 2	PINASSIGN6	Table 103
CTIN_3	I	any	SCT input 3	PINASSIGN6	Table 103
CTOUT_0	O	any	SCT output 0	PINASSIGN6	Table 103
CTOUT_1	O	any	SCT output 1	PINASSIGN7	Table 104
CTOUT_2	O	any	SCT output 2	PINASSIGN7	Table 104
CTOUT_3	O	any	SCT output 3	PINASSIGN7	Table 104

10.5 General description

The State Configurable Timer (SCT) allows a wide variety of timing, counting, output modulation, and input capture operations.

The most basic user-programmable option is whether a SCT operates as two 16-bit counters or a unified 32-bit counter. In the two-counter case, in addition to the counter value the following operational elements are independent for each half:

- State variable
- Limit, halt, stop, and start conditions
- Values of Match/Capture registers, plus reload or capture control values

In the two-counter case, the following operational elements are global to the SCT:

- Clock selection
- Inputs
- Events
- Outputs
- Interrupts

Events, outputs, and interrupts can use match conditions from either counter.

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

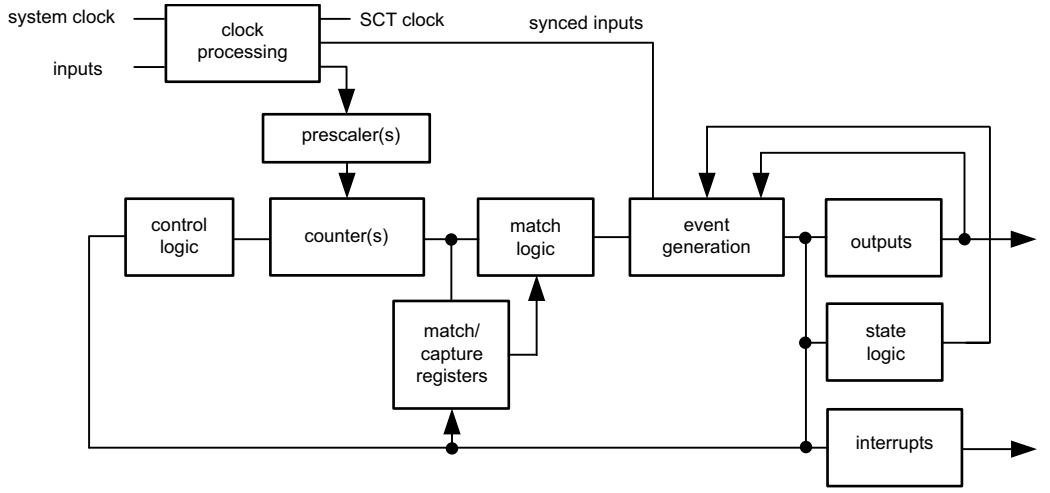


Fig 14. SCT block diagram

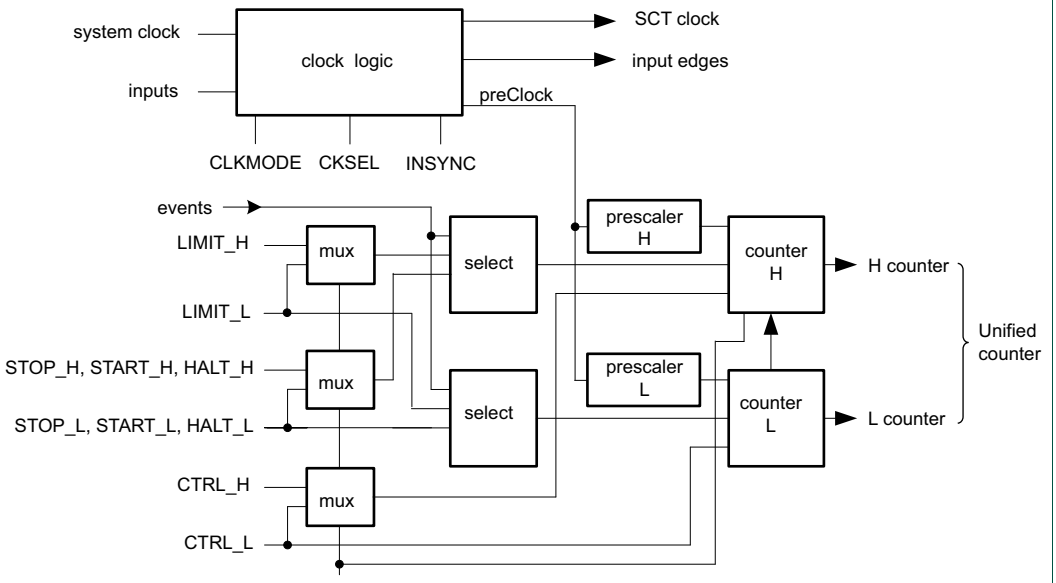


Fig 15. SCT counter and select logic

10.6 Register description

The register addresses of the State Configurable Timer are shown in Table 108. For most of the SCT registers, the register function depends on the setting of certain other register bits:

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

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

Table 108. Register overview: State Configurable Timer (base address 0x5000 4000)

Name	Access	Address offset	Description	Reset value	Reference
CONFIG	R/W	0x000	SCT configuration register	0x0000 7E00	Table 109
CTRL	R/W	0x004	SCT control register	0x0004 0004	Table 110
CTRL_L	R/W	0x004	SCT control register low counter 16-bit	-	Table 110
CTRL_H	R/W	0x006	SCT control register high counter 16-bit	-	Table 110
LIMIT	R/W	0x008	SCT limit register	0x0000 0000	Table 111
LIMIT_L	R/W	0x008	SCT limit register low counter 16-bit	-	Table 111
LIMIT_H	R/W	0x00A	SCT limit register high counter 16-bit	-	Table 111
HALT	R/W	0x00C	SCT halt condition register	0x0000 0000	Table 112
HALT_L	R/W	0x00C	SCT halt condition register low counter 16-bit	-	Table 112
HALT_H	R/W	0x00E	SCT halt condition register high counter 16-bit	-	Table 112
STOP	R/W	0x010	SCT stop condition register	0x0000 0000	Table 113
STOP_L	R/W	0x010	SCT stop condition register low counter 16-bit	-	Table 113
STOP_H	R/W	0x012	SCT stop condition register high counter 16-bit	-	Table 113
START	R/W	0x014	SCT start condition register	0x0000 0000	Table 114
START_L	R/W	0x014	SCT start condition register low counter 16-bit	-	Table 114
START_H	R/W	0x016	SCT start condition register high counter 16-bit	-	Table 114
-	-	0x018 - 0x03C	Reserved		-
COUNT	R/W	0x040	SCT counter register	0x0000 0000	Table 115
COUNT_L	R/W	0x040	SCT counter register low counter 16-bit	-	Table 115
COUNT_H	R/W	0x042	SCT counter register high counter 16-bit	-	Table 115
STATE	R/W	0x044	SCT state register	0x0000 0000	Table 116
STATE_L	R/W	0x044	SCT state register low counter 16-bit	-	Table 116
STATE_H	R/W	0x046	SCT state register high counter 16-bit	-	Table 116
INPUT	RO	0x048	SCT input register	0x0000 0000	Table 117
REGMODE	R/W	0x04C	SCT match/capture registers mode register	0x0000 0000	Table 118
REGMODE_L	R/W	0x04C	SCT match/capture registers mode register low counter 16-bit	-	Table 118
REGMODE_H	R/W	0x04E	SCT match/capture registers mode register high counter 16-bit	-	Table 118
OUTPUT	R/W	0x050	SCT output register	0x0000 0000	Table 119
OUTPUTDIRCTRL	R/W	0x054	SCT output counter direction control register	0x0000 0000	Table 120
RES	R/W	0x058	SCT conflict resolution register	0x0000 0000	Table 121
-	-	0x05C	-	-	-
-	-	0x060	-	-	-
-	-	0x064 - 0x0EC	Reserved	-	-
EVEN	R/W	0x0F0	SCT event enable register	0x0000 0000	Table 122
EVFLAG	R/W	0x0F4	SCT event flag register	0x0000 0000	Table 123
CONEN	R/W	0x0F8	SCT conflict enable register	0x0000 0000	Table 124
CONFLAG	R/W	0x0FC	SCT conflict flag register	0x0000 0000	Table 125

Table 108. Register overview: State Configurable Timer (base address 0x5000 4000) *…continued*

Name	Access	Address offset	Description	Reset value	Reference
MATCH0 to MATCH4	R/W	0x100 to 0x110	SCT match value register of match channels 0 to 4; REGMOD0 to REGMODE4 = 0	0x0000 0000	Table 125
MATCH_L0 to MATCH_L4	R/W	0x100 to 0x110	SCT match value register of match channels 0 to 4; low counter 16-bit; REGMOD0_L to REGMODE4_L = 0	-	Table 125
MATCH_H0 to MATCH_H4	R/W	0x102 to 0x112	SCT match value register of match channels 0 to 4; high counter 16-bit; REGMOD0_H to REGMODE4_H = 0	-	Table 125
CAP0 to CAP4		0x100 to 0x110	SCT capture register of capture channel 0 to 4; REGMOD0 to REGMODE4 = 1	0x0000 0000	Table 127
CAP_L0 to CAP_L4		0x100 to 0x110	SCT capture register of capture channel 0 to 4; low counter 16-bit; REGMOD0_L to REGMODE4_L = 1	-	Table 127
CAP_H0 to CAP_H4		0x102 to 0x13E	SCT capture register of capture channel 0 to 4; high counter 16-bit; REGMOD0_H to REGMODE4_H = 1	-	Table 127
MATCHREL0 to MATCHREL4	R/W	0x200 to 0x210	SCT match reload value register 0 to 4 REGMOD0 = 0 to REGMODE4 = 0	0x0000 0000	Table 128
MATCHREL_L0 to MATCHREL_L4	R/W	0x200 to 0x210	SCT match reload value register 0 to 4; low counter 16-bit; REGMOD0_L = 0 to REGMODE4_L = 0	-	Table 128
MATCHREL_H0 to MATCHREL_H4	R/W	0x202 to 0x212	SCT match reload value register 0 to 4; high counter 16-bit; REGMOD0_H = 0 to REGMODE4_H = 0	-	Table 128
CAPCTRL0 to CAPCTRL4		0x200 to 0x210	SCT capture control register 0 to 4; REGMOD0 = 1 to REGMODE4 = 1	0x0000 0000	Table 129
CAPCTRL_L0 to CAPCTRL_L4		0x200 to 0x210	SCT capture control register 0 to 4; low counter 16-bit; REGMOD0_L = 1 to REGMODE4_L = 1	-	Table 129
CAPCTRL_H0 to CAPCTRL_H4		0x202 to 0x212	SCT capture control register 0 to 4; high counter 16-bit; REGMOD0 = 1 to REGMODE4 = 1	-	Table 129
EV0_STATE	R/W	0x300	SCT event 0 state register	0x0000 0000	Table 130
EV0_CTRL	R/W	0x304	SCT event 0 control register	0x0000 0000	Table 131
EV1_STATE	R/W	0x308	SCT event 1 state register	0x0000 0000	Table 130
EV1_CTRL	R/W	0x30C	SCT event 1 control register	0x0000 0000	Table 131
EV2_STATE	R/W	0x310	SCT event 2 state register	0x0000 0000	Table 130
EV2_CTRL	R/W	0x314	SCT event 2 control register	0x0000 0000	Table 131
EV3_STATE	R/W	0x318	SCT event 3 state register	0x0000 0000	Table 130
EV3_CTRL	R/W	0x31C	SCT event 3 control register	0x0000 0000	Table 131
EV4_STATE	R/W	0x320	SCT event 4 state register	0x0000 0000	Table 130
EV4_CTRL	R/W	0x324	SCT event 4 control register	0x0000 0000	Table 131
EV5_STATE	R/W	0x328	SCT event 5 state register	0x0000 0000	Table 130
EV5_CTRL	R/W	0x32C	SCT event 5 control register	0x0000 0000	Table 131
OUT0_SET	R/W	0x500	SCT output 0 set register	0x0000 0000	Table 132
OUT0_CLR	R/W	0x504	SCT output 0 clear register	0x0000 0000	Table 133
OUT1_SET	R/W	0x508	SCT output 1 set register	0x0000 0000	Table 132

Table 108. Register overview: State Configurable Timer (base address 0x5000 4000) *…continued*

Name	Access	Address offset	Description	Reset value	Reference
OUT1_CLR	R/W	0x50C	SCT output 1 clear register	0x0000 0000	Table 133
OUT2_SET	R/W	0x510	SCT output 2 set register	0x0000 0000	Table 132
OUT2_CLR	R/W	0x514	SCT output 2 clear register	0x0000 0000	Table 133
OUT3_SET	R/W	0x518	SCT output 3 set register	0x0000 0000	Table 132
OUT3_CLR	R/W	0x51C	SCT output 3 clear register	0x0000 0000	Table 133

10.6.1 SCT configuration register

This register configures the overall operation of the SCT. Write to this register before any other registers.

Table 109. SCT configuration register (CONFIG, address 0x5000 4000) bit description

Bit	Symbol	Value	Description	Reset value
0	UNIFY		SCT operation	0
		0	16-bit. The SCT operates as two 16-bit counters named L and H.	
		1	32-bit. The SCT operates as a unified 32-bit counter.	
2:1	CLKMODE		SCT clock mode	0
		0x0	Bus clock. The bus clock clocks the SCT and prescalers.	
		0x1	Prescaled bus clock. The SCT clock is the bus clock, but the prescalers are enabled to count only when sampling of the input selected by the CKSEL field finds the selected edge. The minimum pulse width on the clock input is 1 bus clock period. This mode is the high-performance sampled-clock mode.	
		0x2	Input. The input selected by CKSEL clocks the SCT and prescalers. The input is synchronized to the bus clock and possibly inverted. The minimum pulse width on the clock input is 1 bus clock period. This mode is the low-power sampled-clock mode.	
		0x3	Reserved.	
6:3	CKSEL		SCT clock select. All other values are reserved.	0
		0x0	Input 0 rising edges.	
		0x1	Input 0 falling edges.	
		0x2	Input 1 rising edges.	
		0x3	Input 1 falling edges.	
		0x4	Input 2 rising edges.	
		0x5	Input 2 falling edges.	
		0x6	Input 3 rising edges.	
		0x7	Input 3 falling edges.	
7	NORELAOD_L	-	A 1 in this bit prevents the lower match registers from being reloaded from their respective reload registers. Software can write to set or clear this bit at any time. This bit applies to both the higher and lower registers when the UNIFY bit is set.	0
8	NORELOAD_H	-	A 1 in this bit prevents the higher match registers from being reloaded from their respective reload registers. Software can write to set or clear this bit at any time. This bit is not used when the UNIFY bit is set.	0

Table 109. SCT configuration register (CONFIG, address 0x5000 4000) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
16:9	INSYNC	-	Synchronization for input N (bit 9 = input 0, bit 10 = input 1,..., bit 16 = input 7). A 1 in one of these bits subjects the corresponding input to synchronization to the SCT clock, before it is used to create an event. If an input is synchronous to the SCT clock, keep its bit 0 for faster response. When the CLKMODE field is 1x, the bit in this field, corresponding to the input selected by the CKSEL field, is not used.	1
17	AUTOLIMIT_L	-	A one in this bit causes a match on match register 0 to be treated as a de-facto LIMIT condition without the need to define an associated event. As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in uni-directional mode or to change the direction of count in bi-directional mode. Software can write to set or clear this bit at any time. This bit applies to both the higher and lower registers when the UNIFY bit is set.	
18	AUTOLIMIT_H	-	A one in this bit will cause a match on match register 0 to be treated as a de-facto LIMIT condition without the need to define an associated event. As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in uni-directional mode or to change the direction of count in bi-directional mode. Software can write to set or clear this bit at any time. This bit is not used when the UNIFY bit is set.	
31:19	-		Reserved	-

10.6.2 SCT control register

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers CTRL_L and CTRL_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

All bits in this register can be written to when the counter is stopped or halted. When the counter is running, the only bits that can be written are STOP or HALT. (Other bits can be written in a subsequent write after HALT is set to 1.)

Table 110. SCT control register (CTRL, address 0x5000 4004) bit description

Bit	Symbol	Value	Description	Reset value
0	DOWN_L	-	This bit is 1 when the L or unified counter is counting down. Hardware sets this bit when the counter limit is reached and BIDIR is 1. Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0.	0
1	STOP_L	-	When this bit is 1 and HALT is 0, the L or unified counter does not run, but I/O events related to the counter can occur. If such an event matches the mask in the Start register, this bit is cleared and counting resumes.	0
2	HALT_L	-	When this bit is 1, the L or unified counter does not run and no events can occur. A reset sets this bit. When the HALT_L bit is one, the STOP_L bit is cleared. If you want to remove the halt condition and keep the SCT in the stop condition (not running), then you can change the halt and stop condition with one single write to this register. **Remark:** Once set, only software can clear this bit to restore counter operation.	1

Table 110. SCT control register (CTRL, address 0x5000 4004) bit description

Bit	Symbol	Value	Description	Reset value
3	CLRCTR_L	-	Writing a 1 to this bit clears the L or unified counter. This bit always reads as 0.	0
4	BIDIR_L		L or unified counter direction select	0
		0	Up. The counter counts up to its limit condition, then is cleared to zero.	
		1	Bidirectional. The counter counts up to its limit, then counts down to a limit condition or to 0.	
12:5	PRE_L	-	Specifies the factor by which the SCT clock is prescaled to produce the L or unified counter clock. The counter clock is clocked at the rate of the SCT clock divided by PRE_L+1. **Remark:** Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.	0
15:13	-		Reserved	
16	DOWN_H	-	This bit is 1 when the H counter is counting down. Hardware sets this bit when the counter limit is reached and BIDIR is 1. Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0.	0
17	STOP_H	-	When this bit is 1 and HALT is 0, the H counter does not run, but I/O events related to the counter can occur. If such an event matches the mask in the Start register, this bit is cleared and counting resumes.	0
18	HALT_H	-	When this bit is 1, the H counter does not run and no events can occur. A reset sets this bit. When the HALT_H bit is one, the STOP_H bit is cleared. If you want to remove the halt condition and keep the SCT in the stop condition (not running), then you can change the halt and stop condition with one single write to this register. **Remark:** Once set, this bit can only be cleared by software to restore counter operation.	1
19	CLRCTR_H	-	Writing a 1 to this bit clears the H counter. This bit always reads as 0.	0
20	BIDIR_H		Direction select	0
		0	Up. The H counter counts up to its limit condition, then is cleared to zero.	
		1	Bidirectional. The H counter counts up to its limit, then counts down to a limit condition or to 0.	
28:21	PRE_H	-	Specifies the factor by which the SCT clock is prescaled to produce the H counter clock. The counter clock is clocked at the rate of the SCT clock divided by PRELH+1. **Remark:** Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.	0
31:29	-		Reserved	

10.6.3 SCT limit register

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers LIMIT_L and LIMIT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

The bits in this register set which events act as counter limits. When a limit event occurs, the counter is cleared to zero in unidirectional mode or changes the direction of count in bidirectional mode. When the counter reaches all ones, this state is always treated as a limit event, and the counter is cleared in unidirectional mode or, in bidirectional mode, begins counting down on the next clock edge - even if no limit event as defined by the SCT limit register has occurred.

Note that in addition to using this register to specify events that serve as limits, it is also possible to automatically cause a limit condition whenever a match register 0 match occurs. This eliminates the need to define an event for the sole purpose of creating a limit. The AUTOLIMITL and AUTOLIMITH bits in the configuration register enable/disable this feature (see Table 109).

Table 111. SCT limit register (LIMIT, address 0x5000 4008) bit description

Bit	Symbol	Description	Reset value
5:0	LIMMSK_L	If bit n is one, event n is used as a counter limit for the L or unified counter (event 0 = bit 0, event 1 = bit 1, event 5 = bit 5).	0
15:6	-	Reserved.	-
21:16	LIMMSK_H	If bit n is one, event n is used as a counter limit for the H counter (event 0 = bit 16, event 1 = bit 17, event 5 = bit 21).	0
31:22	-	Reserved.	-

10.6.4 SCT halt condition register

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers HALT_L and HALT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Remark: Any event halting the counter disables its operation until software clears the HALT bit (or bits) in the CTRL register (Table 110).

Table 112. SCT halt condition register (HALT, address 0x5004 400C) bit description

Bit	Symbol	Description	Reset value
5:0	HALTMSK_L	If bit n is one, event n sets the HALT_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, event 5 = bit 5).	0
15:6	-	Reserved.	-
21:16	HALTMSK_H	If bit n is one, event n sets the HALT_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, event 5 = bit 21).	0
31:22	-	Reserved.	-

10.6.5 SCT stop condition register

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers STOPT_L and STOP_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 113. SCT stop condition register (STOP, address 0x5000 4010) bit description

Bit	Symbol	Description	Reset value
5:0	STOPMSK_L	If bit n is one, event n sets the STOP_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, event 5 = bit 5).	0
15:6	-	Reserved.	-
21:16	STOPMSK_H	If bit n is one, event n sets the STOP_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, event 5 = bit 21).	0
31:22	-	Reserved.	-

10.6.6 SCT start condition register

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers START_L and START_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

The bits in this register select which events, if any, clear the STOP bit in the Control register. (Since no events can occur when HALT is 1, only software can clear the HALT bit by writing the Control register.)

Table 114. SCT start condition register (START, address 0x5000 4014) bit description

Bit	Symbol	Description	Reset value
5:0	STARTMSK_L	If bit n is one, event n clears the STOP_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, event 5 = bit 5).	0
15:6	-	Reserved.	-
21:16	STARTMSK_H	If bit n is one, event n clears the STOP_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, event 5 = bit 21).	0
31:22	-	Reserved.	-

10.6.7 SCT counter register

If UNIFY = 1 in the CONFIG register, the counter is a unified 32-bit register and both the _L and _H bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers COUNT_L and COUNT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation. In this case, the L and H registers count independently under the control of the other registers.

Attempting to write a counter while it is running does not affect the counter but produces a bus error. Software can read the counter registers at any time.

Table 115. SCT counter register (COUNT, address 0x5000 4040) bit description

Bit	Symbol	Description	Reset value
15:0	CTR_L	When UNIFY = 0, read or write the 16-bit L counter value. When UNIFY = 1, read or write the lower 16 bits of the 32-bit unified counter.	0
31:16	CTR_H	When UNIFY = 0, read or write the 16-bit H counter value. When UNIFY = 1, read or write the upper 16 bits of the 32-bit unified counter.	0

10.6.8 SCT state register

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers STATE_L and STATE_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Software can read the state associated with a counter at any time. Writing the state is only allowed when the counter HALT bit is 1; when HALT is 0, a write attempt does not change the state and results in a bus error.

The state variable is the main feature that distinguishes the SCT from other counter/timer/PWM blocks. Events can be made to occur only in certain states. Events, in turn, can perform the following actions:

- set and clear outputs
- limit, stop, and start the counter
- cause interrupts
- modify the state variable

The value of a state variable is completely under the control of the application. If an application does not use states, the value of the state variable remains zero, which is the default value.

A state variable can be used to track and control multiple cycles of the associated counter in any desired operational sequence. The state variable is logically associated with a state machine diagram which represents the SCT configuration. See Section 10.6.22 and 10.6.23 for more about the relationship between states and events.

The STATELD/STADEV fields in the event control registers of all defined events set all possible values for the state variable. The change of the state variable during multiple counter cycles reflects how the associated state machine moves from one state to the next.

Table 116. SCT state register (STATE, address 0x5000 4044) bit description

Bit	Symbol	Description	Reset value
4:0	STATE_L	State variable.	0
15:5	-	Reserved.	-
20:16	STATE_H	State variable.	0
31:21	-	Reserved.	

10.6.9 SCT input register

Software can read the state of the SCT inputs in this read-only register in two slightly different forms. The only situation in which these values are different is if CLKMODE = 2 in the CONFIG register.

Table 117. SCT input register (INPUT, address 0x5000 4048) bit description

Bit	Symbol	Description	Reset value
0	AIN0	Real-time status of input 0.	pin
1	AIN1	Real-time status of input 1.	pin
2	AIN2	Real-time status of input 2.	pin
3	AIN3	Real-time status of input 3.	pin
15:4	-	Reserved.	-
16	SIN0	Input 0 state synchronized to the SCT clock.	-
17	SIN1	Input 1 state synchronized to the SCT clock.	-
18	SIN2	Input 2 state synchronized to the SCT clock.	-
19	SIN3	Input 3 state synchronized to the SCT clock.	-
31:20	-	Reserved	-

10.6.10 SCT match/capture registers mode register

If UNIFY = 1 in the CONFIG register, only the _L bits of this register are used. The L bits control whether each set of match/capture registers operates as unified 32-bit capture/match registers.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers REGMODE_L and REGMODE_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.The _L bits/registers control the L match/capture registers, and the _H bits/registers control the H match/capture registers.

The SCT contains 5 Match/Capture register pairs. The Register Mode register selects whether each register pair acts as a Match register (see Section 10.6.18) or as a Capture register (see Section 10.6.19). Each Match/Capture register has an accompanying register which serves as a Reload register when the register is used as a Match register (Section 10.6.20) or as a Capture-Control register when the register is used as a capture register (Section 10.6.21). REGMODE_H is used only when the UNIFY bit is 0.

Table 118. SCT match/capture registers mode register (REGMODE, address 0x5000 404C) bit description

Bit	Symbol	Description	Reset value
4:0	REGMOD_L	Each bit controls one pair of match/capture registers (register 0 = bit 0, register 1 = bit 1,..., register 4 = bit 4). 0 = registers operate as match registers. 1 = registers operate as capture registers.	0
15:5	-	Reserved.	-
20:16	REGMOD_H	Each bit controls one pair of match/capture registers (register 0 = bit 16, register 1 = bit 17,..., register 4 = bit 20). 0 = registers operate as match registers. 1 = registers operate as capture registers.	0
31:21	-	Reserved.	-

10.6.11 SCT output register

The SCT supports 4 outputs, each of which has a corresponding bit in this register.

Software can write to any of the output registers when both counters are halted to control the outputs directly. Writing to this register when either counter is stopped or running does not affect the outputs and results in an bus error.

Software can read this register at any time to sense the state of the outputs.

Table 119. SCT output register (OUTPUT, address 0x5000 4050) bit description

Bit	Symbol	Description	Reset value
3:0	OUT	Writing a 1 to bit n makes the corresponding output HIGH. 0 makes the corresponding output LOW (output 0 = bit 0, output 1 = bit 1,..., output 3 = bit 3).	0
31:4	-	Reserved	

10.6.12 SCT bidirectional output control register

This register specifies (for each output) the impact of the counting direction on the meaning of set and clear operations on the output (see Section 10.6.24 and Section 10.6.25).

Table 120. SCT bidirectional output control register (OUTPUTDIRCTRL, address 0x5000 4054) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SETCLR0		Set/clear operation on output 0. Value 0x3 is reserved. Do not program this value.	0
		0x0	Any. Set and clear do not depend on any counter.	
		0x1	L counting down. Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	H counting down. Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	

Table 120. SCT bidirectional output control register (OUTPUTDIRCTRL, address 0x5000 4054) bit description

Bit	Symbol	Value	Description	Reset value
3:2	SETCLR1		Set/clear operation on output 1. Value 0x3 is reserved. Do not program this value.	0
		0x0	Any. Set and clear do not depend on any counter.	
		0x1	L counting down. Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	H counting down. Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
5:4	SETCLR2		Set/clear operation on output 2. Value 0x3 is reserved. Do not program this value.	0
		0x0	Any. Set and clear do not depend on any counter.	
		0x1	L counting down. Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	H counting down. Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
7:6	SETCLR3		Set/clear operation on output 3. Value 0x3 is reserved. Do not program this value.	0
		0x0	Any. Set and clear do not depend on any counter.	
		0x1	L counting down. Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	H counting down. Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
31:8	-		Reserved	-

10.6.13 SCT conflict resolution register

The registers OUTn_SETn (Section 10.6.24) and OUTnCLRn (Section 10.6.25) allow both setting and clearing to be indicated for an output in the same clock cycle, even for the same event. This SCT conflict resolution register resolves this conflict.

To enable an event to toggle an output, set the OnRES value to 0x3 in this register, and set the event bits in both the Set and Clear registers.

Table 121. SCT conflict resolution register (RES, address 0x5000 4058) bit description

Bit	Symbol	Value	Description	Reset value
1:0	O0RES		Effect of simultaneous set and clear on output 0.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR0 field).	
		0x2	Clear output (or set based on the SETCLR0 field).	
		0x3	Toggle output.	
3:2	O1RES		Effect of simultaneous set and clear on output 1.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR1 field).	
		0x2	Clear output (or set based on the SETCLR1 field).	
		0x3	Toggle output.	

Table 121. SCT conflict resolution register (RES, address 0x5000 4058) bit description

Bit	Symbol	Value	Description	Reset value
5:4	O2RES		Effect of simultaneous set and clear on output 2.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR2 field).	
		0x2	Clear output n (or set based on the SETCLR2 field).	
		0x3	Toggle output.	
7:6	O3RES		Effect of simultaneous set and clear on output 3.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR3 field).	
		0x2	Clear output (or set based on the SETCLR3 field).	
		0x3	Toggle output.	
31:8	-	-	Reserved	-

10.6.14 SCT flag enable register

This register enables flags to request an interrupt if the FLAGn bit in the SCT event flag register (Section 10.6.15) is also set.

Table 122. SCT flag enable register (EVEN, address 0x5000 40F0) bit description

Bit	Symbol	Description	Reset value
5:0	IEN	The SCT requests an interrupt when bit n of this register and the event flag register are both one (event 0 = bit 0, event 1 = bit 1,..., event 5 = bit 5).	0
31:6	-	Reserved	

10.6.15 SCT event flag register

This register records events. Writing ones to this register clears the corresponding flags and negates the SCT interrupt request if all enabled Flag bits are zero.

Table 123. SCT event flag register (EVFLAG, address 0x5000 40F4) bit description

Bit	Symbol	Description	Reset value
5:0	FLAG	Bit n is one if event n has occurred since reset or a 1 was last written to this bit (event 0 = bit 0, event 1 = bit 1,..., event 5 = bit 5).	0
31:6	-	Reserved	-

10.6.16 SCT conflict enable register

This register enables the “no change conflict” events specified in the SCT conflict resolution register to request an IRQ.

Table 124. SCT conflict enable register (CONEN, address 0x5000 40F8) bit description

Bit	Symbol	Description	Reset value
3:0	NCEN	The SCT requests interrupt when bit n of this register and the SCT conflict flag register are both one (output 0 = bit 0, output 1 = bit 1,..., output 3 = bit 3).	0
31:4	-	Reserved	

10.6.17 SCT conflict flag register

This register records interrupt-enabled no-change conflict events and provides details of a bus error. Writing ones to the NCFLAG bits clears the corresponding read bits and negates the SCT interrupt request if all enabled Flag bits are zero.

Table 125. SCT conflict flag register (CONFLAG, address 0x5000 40FC) bit description

Bit	Symbol	Description	Reset value
3:0	NCFLAG	Bit n is one if a no-change conflict event occurred on output n since reset or a 1 was last written to this bit (output 0 = bit 0, output 1 = bit 1,..., output 3 = bit 3).	0
29:4	-	Reserved.	-
30	BUSERRL	The most recent bus error from this SCT involved writing CTR L/Unified, STATE L/Unified, MATCH L/Unified, or the Output register when the L/U counter was not halted. A word write to certain L and H registers can be half successful and half unsuccessful.	0
31	BUSERRH	The most recent bus error from this SCT involved writing CTR H, STATE H, MATCH H, or the Output register when the H counter was not halted.	0

10.6.18 SCT match registers 0 to 4 (REGMODEn bit = 0)

Match registers are compared to the counters to help create events. When the UNIFY bit is 0, the L and H registers are independently compared to the L and H counters. When UNIFY is 1, the L and H registers hold a 32-bit value that is compared to the unified counter. A Match can only occur in a clock in which the counter is running (STOP and HALT are both 0).

Match registers can be read at any time. Writing to a Match register while the associated counter is running does not affect the Match register and results in a bus error. Match events occur in the SCT clock in which the counter is (or would be) incremented to the next value. When a Match event limits its counter as described in Section 10.6.3, the value in the Match register is the last value of the counter before it is cleared to zero (or decremented if BIDIR is 1).

There is no “write-through” from Reload registers to Match registers. Before starting a counter, software can write one value to the Match register used in the first cycle of the counter and a different value to the corresponding Match Reload register used in the second cycle.

Table 126. SCT match registers 0 to 4 (MATCH[0:4], address 0x5000 4100 (MATCH0) to 0x5000 4110 (MATCH4)) bit description (REGMODEn bit = 0)

Bit	Symbol	Description	Reset value
15:0	VALMATCH_L	When UNIFY = 0, read or write the 16-bit value to be compared to the L counter. When UNIFY = 1, read or write the lower 16 bits of the 32-bit value to be compared to the unified counter.	0
31:16	VALMATCH_H	When UNIFY = 0, read or write the 16-bit value to be compared to the H counter. When UNIFY = 1, read or write the upper 16 bits of the 32-bit value to be compared to the unified counter.	0

10.6.19 SCT capture registers 0 to 4 (REGMODEn bit = 1)

These registers allow software to read the counter values at which the event selected by the corresponding Capture Control registers occurred.

Table 127. SCT capture registers 0 to 4 (CAP[0:4], address 0x5000 4100 (CAP0) to 0x5000 4110 (CAP4)) bit description (REGMODEn bit = 1)

Bit	Symbol	Description	Reset value
15:0	VALCAP_L	When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the lower 16 bits of the 32-bit value at which this register was last captured.	0
31:16	VALCAP_H	When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the upper 16 bits of the 32-bit value at which this register was last captured.	0

10.6.20 SCT match reload registers 0 to 4 (REGMODEn bit = 0)

A Match register (L, H, or unified 32-bit) is loaded from the corresponding Reload register when BIDIR is 0 and the counter reaches its limit condition, or when BIDIR is 1 and the counter reaches 0.

Table 128. SCT match reload registers 0 to 4 (MATCHREL[0:4], address 0x5000 4200 (MATCHREL0) to 0x5000 4210 (MATCHREL4) bit description (REGMODEn bit = 0)

Bit	Symbol	Description	Reset value
15:0	RELOAD_L	When UNIFY = 0, read or write the 16-bit value to be loaded into the SCTMATCHn_L register. When UNIFY = 1, read or write the lower 16 bits of the 32-bit value to be loaded into the MATCHn register.	0
31:16	RELOAD_H	When UNIFY = 0, read or write the 16-bit to be loaded into the MATCHn_H register. When UNIFY = 1, read or write the upper 16 bits of the 32-bit value to be loaded into the MATCHn register.	0

10.6.21 SCT capture control registers 0 to 4 (REGMODEn bit = 1)

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers CAPCTRLn_L and CAPCTRLn_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Each Capture Control register (L, H, or unified 32-bit) controls which events load the corresponding Capture register from the counter.

Table 129. SCT capture control registers 0 to 4 (CAPCTRL[0:4], address 0x5000 4200 (CAPCTRL0) to 0x5000 4210 (CAPCTRL4)) bit description (REGMODEn bit = 1)

Bit	Symbol	Description	Reset value
5:0	CAPCONm_L	If bit m is one, event m causes the CAPn_L (UNIFY = 0) or the CAPn (UNIFY = 1) register to be loaded (event 0 = bit 0, event 1 = bit 1,..., event 5 = bit 5).	0
15:6	-	Reserved.	-
21:16	CAPCONm_H	If bit m is one, event m causes the CAPn_H (UNIFY = 0) register to be loaded (event 0 = bit 16, event 1 = bit 17,..., event 5 = bit 21).	0
31:22	-	Reserved.	-

10.6.22 SCT event state mask registers 0 to 5

Each event has one associated SCT event state mask register that allow this event to happen in one or more states of the counter selected by the HEVENT bit in the corresponding EVn_CTRL register.

An event n is disabled when its EVn_STATE register contains all zeros, since it is masked regardless of the current state.

In simple applications that do not use states, write 0x01 to this register to enable an event. Since the state always remains at its reset value of 0, writing 0x01 permanently state-enables this event.

Table 130. SCT event state mask registers 0 to 5 (EV[0:5]_STATE, addresses 0x5000 4300 (EV0_STATE) to 0x5000 4328 (EV5_STATE)) bit description

Bit	Symbol	Description	Reset value
1:0	STATEMSKm	If bit m is one, event n (n= 0 to 5) happens in state m of the counter selected by the HEVENT bit (m = state number; state 0 = bit 0, state 1= bit 1).	0
31:2	-	Reserved.	-

10.6.23 SCT event control registers 0 to 5

This register defines the conditions for event n to occur, other than the state variable which is defined by the state mask register. Most events are associated with a particular counter (high, low, or unified), in which case the event can depend on a match to that register. The other possible ingredient of an event is a selected input or output signal.

When the UNIFY bit is 0, each event is associated with a particular counter by the HEVENT bit in its event control register. An event cannot occur when its related counter is halted nor when the current state is not enabled to cause the event as specified in its event mask register. An event is permanently disabled when its event state mask register contains all 0s.

An enabled event can be programmed to occur based on a selected input or output edge or level and/or based on its counter value matching a selected match register (STOP bit = 0). An event can be enabled by the event counter's HALT bit and STATE register. In bi-directional mode, events can also be enabled based on the direction of count.

Each event can modify its counter STATE value. If more than one event associated with the same counter occurs in a given clock cycle, only the state change specified for the highest-numbered event among them takes place. Other actions dictated by any simultaneously occurring events all take place.

Table 131. SCT event control register 0 to 5 (EV[0:5]_CTRL, address 0x5000 4304 (EV0_CTRL) to 0x5000 432C (EV5_CTRL)) bit description

Bit	Symbol	Value	Description	Reset value
3:0	MATCHSEL	-	Selects the Match register associated with this event (if any). A match can occur only when the counter selected by the HEVENT bit is running.	0
4	HEVENT		Select L/H counter. Do not set this bit if UNIFY = 1.	0
		0	L state. Selects the L state and the L match register selected by MATCHSEL.	
		1	H state. Selects the H state and the H match register selected by MATCHSEL.	
5	OUTSEL		Input/output select	0
		0	Input. Selects the inputs elected by IOSEL.	
		1	Output. Selects the outputs selected by IOSEL.	
9:6	IOSEL	-	Selects the input or output signal associated with this event (if any). Do not select an input in this register, if CLKMODE is 1x. In this case the clock input is an implicit ingredient of every event. IOSEL = 0 selects pins CTIN_0 or CTOUT_0, ..., IOSEL = 3 selects pins CTIN_3 or CTOUT_3.	0
11:10	IOCOND		Selects the I/O condition for event n. (The detection of edges on outputs lag the conditions that switch the outputs by one SCT clock). In order to guarantee proper edge/state detection, an input must have a minimum pulse width of at least one SCT clock period .	0
		0x0	LOW	
		0x1	Rise	
		0x2	Fall	
		0x3	HIGH	
13:12	COMBMODE		Selects how the specified match and I/O condition are used and combined.	
		0x0	OR. The event occurs when either the specified match or I/O condition occurs.	
		0x1	MATCH. Uses the specified match only.	
		0x2	IO. Uses the specified I/O condition only.	
		0x3	AND. The event occurs when the specified match and I/O condition occur simultaneously.	
14	STATELD		This bit controls how the STATEV value modifies the state selected by HEVENT when this event is the highest-numbered event occurring for that state.	
		0	Add. STATEV value is added into STATE (the carry-out is ignored).	
		1	Load. STATEV value is loaded into STATE.	
19:15	STATEV		This value is loaded into or added to the state selected by HEVENT, depending on STATELD, when this event is the highest-numbered event occurring for that state. If STATELD and STATEV are both zero, there is no change to the STATE value.	

Table 131. SCT event control register 0 to 5 (EV[0:5]_CTRL, address 0x5000 4304 (EV0_CTRL) to 0x5000 432C (EV5_CTRL)) bit description

Bit	Symbol	Value	Description	Reset value
20	MATCHMEM		If this bit is one and the COMBMODE field specifies a match component to the triggering of this event, then a match is considered to be active whenever the counter value is GREATER THAN OR EQUAL TO the value specified in the match register when counting up, LESS THEN OR EQUAL TO the match value when counting down. If this bit is zero, a match is only be active during the cycle when the counter is equal to the match value.	
22:21	DIRECTION		Direction qualifier for event generation. This field only applies when the counters are operating in BIDIR mode. If BIDIR = 0, the SCT ignores this field. Value 0x3 is reserved.	
		0x0	Direction independent. This event is triggered regardless of the count direction.	
		0x1	Counting up. This event is triggered only during up-counting when BIDIR = 1.	
		0x2	Counting down. This event is triggered only during down-counting when BIDIR = 1.	
31:23	-		Reserved	

10.6.24 SCT output set registers 0 to 3

Each output n has one set register that controls how events affect each output. Whether outputs are set or cleared depends on the setting of the SETCLRn field in the SCT OUTPUTDIRCTRL register.

Table 132. SCT output set register (OUT[0:3]_SET, address 0x5000 4500 (OUT0_SET) to 0x5000 4518 (OUT3_SET)) bit description

Bit	Symbol	Description	Reset value
5:0	SET	A 1 in bit m selects event m to set output n (or clear it if SETCLRn = 0x1 or 0x2) event 0 = bit 0, event 1 = bit 1,..., event 5 = bit 5.	0
31:6	-	Reserved	

10.6.25 SCT output clear registers 0 to 3

Each output n has one clear register that controls how events affect each output. Whether outputs are set or cleared depends on the setting of the SETCLRn field in the OUTPUTDIRCTRL register.

Table 133. SCT output clear register (OUT[0:3]_CLR, address 0x5000 0504 (OUT0_CLR) to 0x5000 051C (OUT3_CLR)) bit description

Bit	Symbol	Description	Reset value
5:0	CLR	A 1 in bit m selects event m to clear output n (or set it if SETCLRn = 0x1 or 0x2) event 0 = bit 0, event 1 = bit 1,..., event 5 = bit 5.	0
31:6	-	Reserved	

10.7 Functional description

10.7.1 Match logic

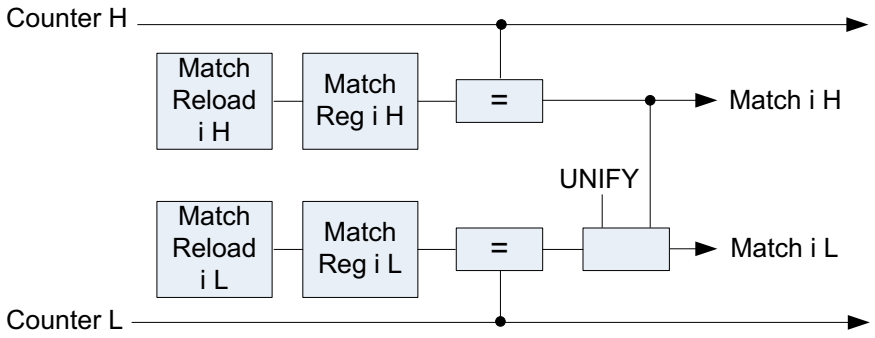


Fig 16. Match logic

10.7.2 Capture logic

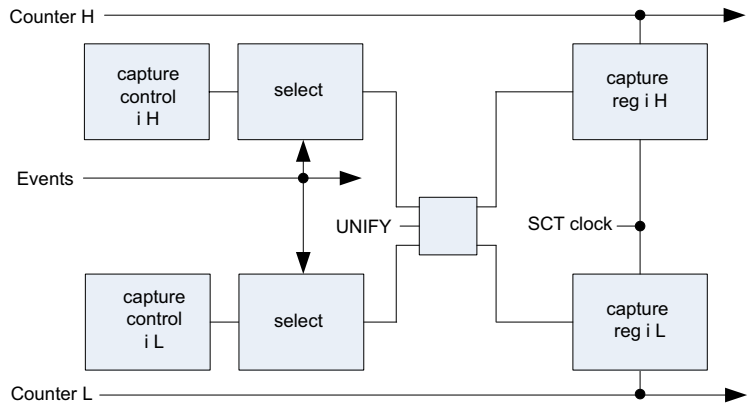


Fig 17. Capture logic

10.7.3 Event selection

State variables allow control of the SCT across more than one cycle of the counter. Counter matches, input/output edges, and state values are combined into a set of general-purpose events that can switch outputs, request interrupts, and change state values.

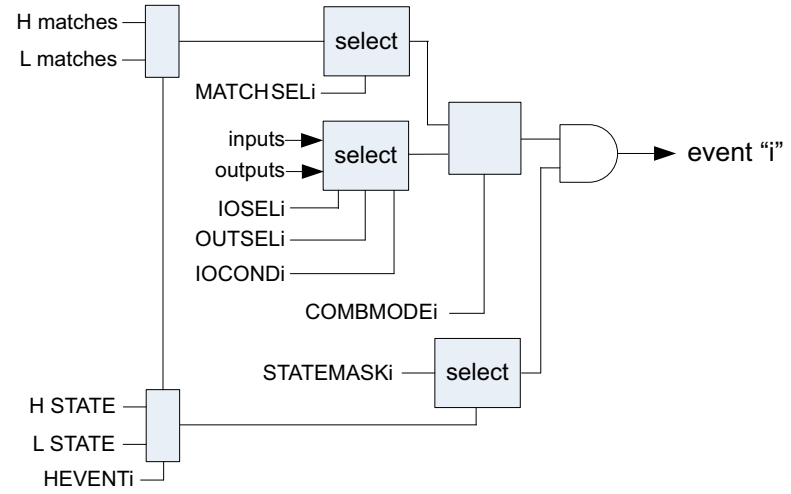


Fig 18. Event selection

10.7.4 Output generation

Figure 19 shows one output slice of the SCT.

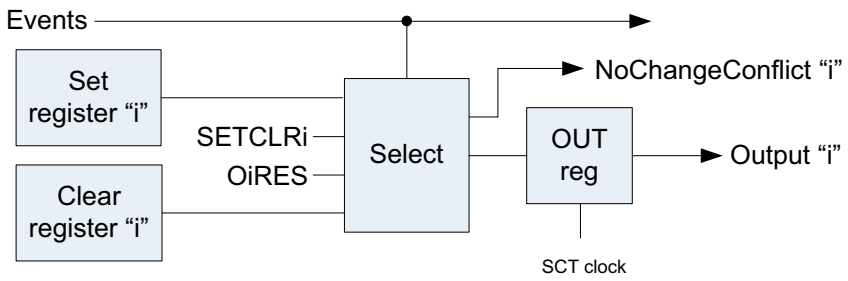


Fig 19. Output slice i

10.7.5 Interrupt generation

The SCT generates one interrupt to the NVIC.

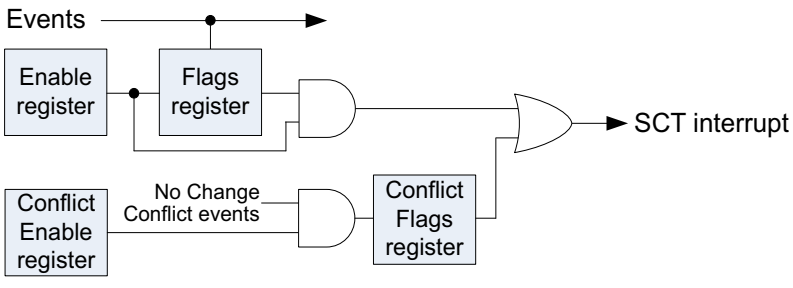


Fig 20. SCT interrupt generation

10.7.6 Clearing the prescaler

When enabled by a non-zero PRE field in the Control register, the prescaler acts as a clock divider for the counter, like a fractional part of the counter value. The prescaler is cleared whenever the counter is cleared or loaded for any of the following reasons:

- Hardware reset
- Software writing to the counter register
- Software writing a 1 to the CLRCTR bit in the control register
- an event selected by a 1 in the counter limit register when BIDIR = 0

When BIDIR is 0, a limit event caused by an I/O signal can clear a non-zero prescaler. However, a limit event caused by a Match only clears a non-zero prescaler in one special case as described Section 10.7.7.

A limit event when BIDIR is 1 does not clear the prescaler. Rather it clears the DOWN bit in the Control register, and decrements the counter on the same clock if the counter is enabled in that clock.

10.7.7 Match vs. I/O events

Counter operation is complicated by the prescaler and by clock mode 01 in which the SCT clock is the bus clock. However, the prescaler and counter are enabled to count only when a selected edge is detected on a clock input.

- The prescaler is enabled when the clock mode is not 01, or when the input edge selected by the CLKSEL field is detected.
- The counter is enabled when the prescaler is enabled, and (PRELIM=0 or the prescaler is equal to the value in PRELIM).

An I/O component of an event can occur in any SCT clock when its counter HALT bit is 0. In general, a Match component of an event can only occur in a UT clock when its counter HALT and STOP bits are both 0 and the counter is enabled.

Table 134 shows when the various kinds of events can occur.

Table 134. Event conditions

COMBMODE	IOMODE	Event can occur on clock:
IO	Any	Event can occur whenever HALT = 0 (type A).
MATCH	Any	Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (type C).
OR	Any	From the IO component: Event can occur whenever HALT = 0 (A). From the match component: Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (C).
AND	LOW or HIGH	Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (C).
AND	RISE or FALL	Event can occur whenever HALT = 0 (A).

10.7.8 SCT operation

In its simplest, single-state configuration, the SCT operates as an event controlled one- or bidirectional counter. Events can be configured to be counter match events, an input or output level, transitions on an input or output pin, or a combination of match and input/output behavior. In response to an event, the SCT output or outputs can transition, or the SCT can perform other actions such as creating an interrupt or starting, stopping, or resetting the counter. Multiple simultaneous actions are allowed for each event. Furthermore, any number of events can trigger one specific action of the SCT.

An action or multiple actions of the SCT uniquely define an event. A state is defined by which events are enabled to trigger an SCT action or actions in any stage of the counter. Events not selected for this state are ignored.

In a multi-state configuration, states change in response to events. A state change is an additional action that the SCT can perform when the event occurs. When an event is configured to change the state, the new state defines a new set of events resulting in different actions of the SCT. Through multiple cycles of the counter, events can change the state multiple times and thus create a large variety of event controlled transitions on the SCT outputs and/or interrupts.

Once configured, the SCT can run continuously without software intervention and can generate multiple output patterns entirely under the control of events.

- To configure the SCT, see Section 10.7.9.
- To start, run, and stop the SCT, see Section 10.7.10.
- To configure the SCT as simple event controlled counter/timer, see Section 10.7.11.

10.7.9 Configure the SCT

To set up the SCT for multiple events and states, perform the following configuration steps:

10.7.9.1 Configure the counter

1. Configure the L and H counters in the CONFIG register by selecting two independent 16-bit counters (L counter and H counter) or one combined 32-bit counter in the UNIFY field.
2. Select the SCT clock source in the CONFIG register (fields CLKMODE and CLKSEL) from any of the inputs or an internal clock.

10.7.9.2 Configure the match and capture registers

1. Select how many match and capture registers the application uses (total of up to 5):
   - In the REGMODE register, select for each of the 5 match/capture register pairs whether the register is used as a match register or capture register.
2. Define match conditions for each match register selected:
   - Each match register MATCH sets one match value, if a 32-bit counter is used, or two match values, if the L and H 16-bit counters are used.
   - Each match reload register MATCHRELOAD sets a reload value that is loaded into the match register when the counter reaches a limit condition or the value 0.

10.7.9.3 Configure events and event responses

1. Define when each event can occur in the following way in the EVn_CTRL registers (up to 6, one register per event):
   - Select whether the event occurs on an input or output changing, on an input or output level, a match condition of the counter, or a combination of match and input/output conditions in field COMBMODE.
   - For a match condition:

     Select the match register that contains the match condition for the event to occur. Enter the number of the selected match register in field MATCHSEL.

     If using L and H counters, define whether the event occurs on matching the L or the H counter in field HEVENT.
   - For an SCT input or output level or transition:

     Select the input number or the output number that is associated with this event in fields IOSEL and OUTSEL.

     Define how the selected input or output triggers the event (edge or level sensitive) in field IOCOND.
2. Define what the effect of each event is on the SCT outputs in the OUTn_SET or OUTn_CLR registers (up to 4 outputs, one register per output):
   - For each SCT output, select which events set or clear this output. More than one event can change the output, and each event can change multiple outputs.
3. Define how each event affects the counter:
   - Set the corresponding event bit in the LIMIT register for the event to set an upper limit for the counter.

     When a limit event occurs in unidirectional mode, the counter is cleared to zero and begins counting up on the next clock edge.

     When a limit event occurs in bidirectional mode, the counter begins to count down from the current value on the next clock edge.
   - Set the corresponding event bit in the HALT register for the event to halt the counter. If the counter is halted, it stops counting and no new events can occur. The counter operation can only be restored by clearing the HALT_L and/or the HALT_H bits in the CTRL register.
   - Set the corresponding event bit in the STOP register for the event to stop the counter. If the counter is stopped, it stops counting. However, an event that is configured as a transition on an input/output can restart the counter.
   - Set the corresponding event bit in the START register for the event to restart the counting. Only events that are defined by an input changing can be used to restart the counter.
4. Define which events contribute to the SCT interrupt:
   - Set the corresponding event bit in the EVEN and the EVFLAG registers to enable the event to contribute to the SCT interrupt.

10.7.9.4 Configure multiple states

1. In the EVn_STATE register for each event (up to 6 events, one register per event), select the state or states (up to 2) in which this event is allowed to occur. Each state can be selected for more than one event.
2. Determine how the event affects the system state:

   In the EVn_CTRL registers (up to 6 events, one register per event), set the new state value in the STATEV field for this event. If the event is the highest numbered in the current state, this value is either added to the existing state value or replaces the existing state value, depending on the field STATELD.

   Remark: If there are higher numbered events in the current state, this event cannot change the state.

   If the STATEV and STATELD values are set to zero, the state does not change.

10.7.9.5 Miscellaneous options

- There are a certain (selectable) number of capture registers. Each capture register can be programmed to capture the counter contents when one or more events occur.
- If the counter is in bidirectional mode, the effect of set and clear of an output can be made to depend on whether the counter is counting up or down by writing to the OUTPUTDIRCTRL register.

10.7.10 Run the SCT

1. Configure the SCT (see Section 10.7.9 “Configure the SCT”).
2. Write to the STATE register to define the initial state. By default the initial state is state 0.
3. To start the SCT, write to the CTRL register:
   - Clear the counters.
   - Clear or set the STOP_L and/or STOP_H bits.

     Remark: The counter starts counting once the STOP bit is cleared as well. If the STOP bit is set, the SCT waits instead for an event to occur that is configured to start the counter.
   - For each counter, select unidirectional or bidirectional counting mode (field BIDIR_L and/or BIDIR_H).
   - Select the prescale factor for the counter clock (CTRL register).
   - Clear the HALT_L and/or HALT_H bit. By default, the counters are halted and no events can occur.
4. To stop the counters by software at any time, stop or halt the counter (write to STOP_L and/or STOP_H bits or HALT_L and/or HALT_H bits in the CTRL register).
   - When the counters are stopped, both an event configured to clear the STOP bit or software writing a zero to the STOP bit can start the counter again.
   - When the counter are halted, only a software write to clear the HALT bit can start the counter again. No events can occur.
   - When the counters are halted, software can set any SCT output HIGH or LOW directly by writing to the OUT register.

The current state can be read at any time by reading the STATE register.

To change the current state by software (that is independently of any event occurring), set the HALT bit and write to the STATE register to change the state value. Writing to the STATE register is only allowed when the counter is halted (the HALT_L and/or HALT_H bits are set) and no events can occur.

10.7.11 Configure the SCT without using states

The SCT can be used as standard counter/timer with external capture inputs and match outputs without using the state logic. To operate the SCT without states, configure the SCT as follows:

- Write zero to the STATE register (zero is the default).
- Write zero to the STATELD and STATEV fields in the EVCTRL registers for each event.
- Write 0x1 to the EVn_STATE register of each event. Writing 0x1 enables the event.

  In effect, the event is allowed to occur in a single state which never changes while the counter is running.

11.1 How to read this chapter

The MRT is available on all LPC800 parts.

11.2 Features

- 31-bit interrupt timer
- Four channels independently counting down from individually set values
- Repeat and one-shot interrupt modes

11.3 Basic configuration

Configure the MRT using the following registers:

- In the SYSAHBCLKCTRL register, set bit 10 (Table 18) to enable the clock to the register interface.
- Clear the MRT reset using the PRESETCTRL register (Table 7).
- The global MRT interrupt is connected to interrupt #10 in the NVIC.

11.4 Pin description

The MRT has no configurable pins.

11.5 General description

The Multi-Rate Timer (MRT) provides a repetitive interrupt timer with four channels. Each channel can be programmed with an independent time interval.

Each channel operates independently from the other channels in one of the following modes:

- Repeat interrupt mode. See Section 11.5.1.
- One-shot interrupt mode. See Section 11.5.2.

The modes for each timer are set in the timer's control register. See Table 138.

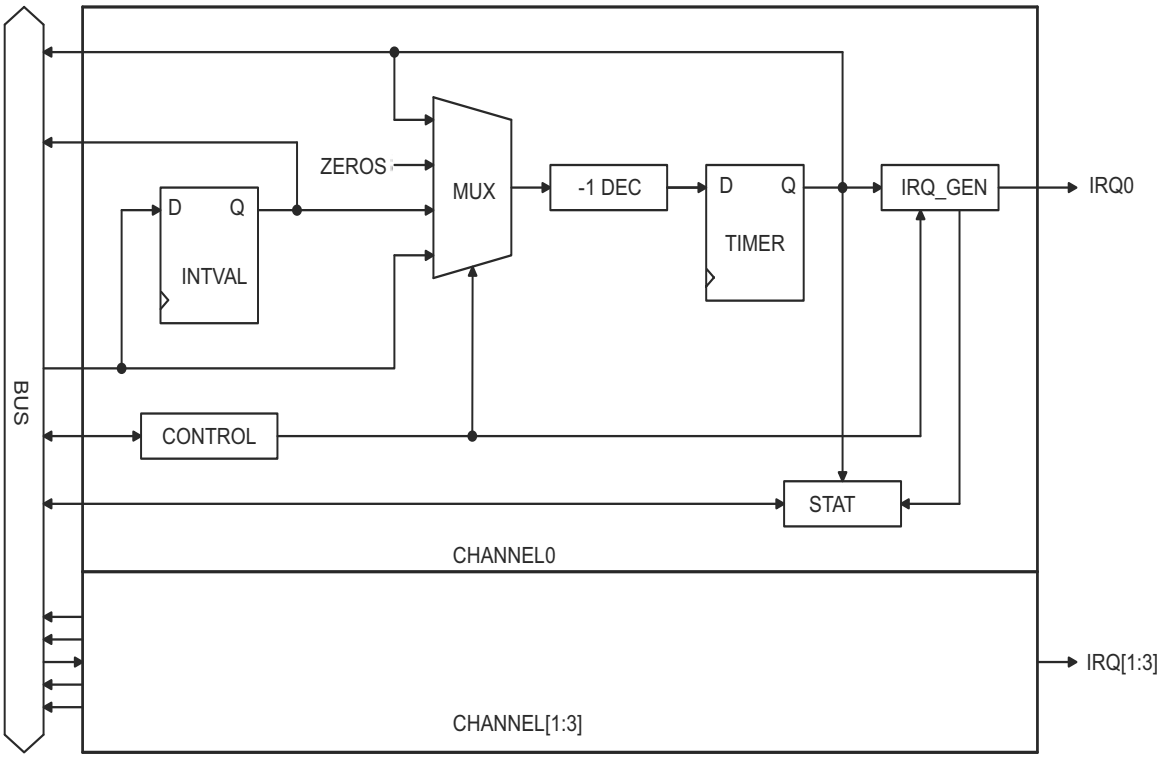


Fig 21. MRT block diagram

11.5.1 Repeat interrupt mode

The repeat interrupt mode generates repeated interrupts after a selected time interval. This mode can be used for software-based PWM or PPM applications.

When the timer n is in idle state, writing a non-zero value IVALUE to the INTVALn register immediately loads the time interval value IVALUE - 1, and the timer begins to count down from this value. When the timer reaches zero, an interrupt is generated, the value in the INTVALn register IVALUE - 1 is reloaded automatically, and the timer starts to count down again.

While the timer is running in repeat interrupt mode, you can perform the following actions:

- Change the interval value on the next timer cycle by writing a new value (>0) to the INTVALn register and setting the LOAD bit to 0. An interrupt is generated when the timer reaches zero. On the next cycle, the timer counts down from the new value.
- Change the interval value on-the-fly immediately by writing a new value (>0) to the INTVALn register and setting the LOAD bit to 1. The timer immediately starts to count down from the new timer interval value. An interrupt is generated when the timer reaches 0.
- Stop the timer at the end of time interval by writing a 0 to the INTVALn register and setting the LOAD bit to 0. An interrupt is generated when the timer reaches zero.
- Stop the timer immediately by writing a 0 to the INTVALn register and setting the LOAD bit to 1. No interrupt is generated when the INTVALn register is written.

11.5.2 One-shot interrupt mode

The one-shot interrupt generates one interrupt after a one-time count. With this mode, you can generate a single interrupt at any point. This mode can be used to introduce a specific delay in a software task.

When the timer is in the idle state, writing a non-zero value IVALUE to the INTVALn register immediately loads the time interval value IVALUE - 1, and the timer starts to count down. When the timer reaches 0, an interrupt is generated and the timer stops and enters the idle state.

While the timer is running in the one-shot interrupt mode, you can perform the following actions:

- Update the INTVALn register with a new time interval value (>0) and set the LOAD bit to 1. The timer immediately reloads the new time interval, and starts counting down from the new value. No interrupt is generated when the TIME_INTVALn register is updated.
- Write a 0 to the INTVALn register and set the LOAD bit to 1. The timer immediately stops counting and moves to the idle state. No interrupt is generated when the INTVALn register is updated.

11.6 Register description

The reset values shown in Table 135 are POR reset values.

Table 135. Register overview: MRT (base address 0x4000 4000)

Name	Access	Address offset	Description	Reset value	Reference
INTVAL0	R/W	0x0	MRT0 Time interval value register. This value is loaded into the TIMER0 register.	0	Table 136
TIMER0	R	0x4	MRT0 Timer register. This register reads the value of the down-counter.	0x7FFF FFFF	Table 137
CTRL0	R/W	0x8	MRT0 Control register. This register controls the MRT0 modes.	0	Table 138
STAT0	R/W	0xC	MRT0 Status register.	0	Table 139
INTVAL1	R/W	0x10	MRT1 Time interval value register. This value is loaded into the TIMER1 register.	0	Table 136
TIMER1	R/W	0x14	MRT1 Timer register. This register reads the value of the down-counter.	0x7FFF FFFF	Table 137
CTRL1	R/W	0x18	MRT1 Control register. This register controls the MRT1 modes.	0	Table 138
STAT1	R/W	0x1C	MRT1 Status register.	0	Table 139
INTVAL2	R/W	0x20	MRT2 Time interval value register. This value is loaded into the TIMER2 register.	0	Table 136
TIMER2	R/W	0x24	MRT2 Timer register. This register reads the value of the down-counter.	0x7FFF FFFF	Table 137
CTRL2	R/W	0x28	MRT2 Control register. This register controls the MRT2 modes.	0	Table 138
STAT2	R/W	0x2C	MRT2 Status register.	0	Table 139

Table 135. Register overview: MRT (base address 0x4000 4000)

Name	Access	Address offset	Description	Reset value	Reference
INTVAL3	R/W	0x30	MRT3 Time interval value register. This value is loaded into the TIMER3 register.	0	Table 136
TIMER3	R/W	0x34	MRT3 Timer register. This register reads the value of the down-counter.	0x7FFF FFFF	Table 137
CTRL3	R/W	0x38	MRT3 Control register. This register controls the MRT modes.	0	Table 138
STAT3	R/W	0x3C	MRT3 Status register.	0	Table 139
IDLE_CH	R	0xF4	Idle channel register. This register returns the number of the first idle channel.	0	Table 140
IRQ_FLAG	R/W	0xF8	Global interrupt flag register	0	Table 141

11.6.1 Time interval register

This register contains the MRT load value and controls how the timer is reloaded. The load value is IVALUE -1.

Table 136. Time interval register (INTVAL[0:3], address 0x4000 4000 (INTVAL0) to 0x4000 4030 (INTVAL3)) bit description

Bit	Symbol	Value	Description	Reset value
30:0	IVALUE		Time interval load value. This value is loaded into the TIMERn register and the MRTn starts counting down from IVALUE -1. If the timer is idle, writing a non-zero value to this bit field starts the timer immediately. If the timer is running, writing a zero to this bit field does the following: • If LOAD = 1, the timer stops immediately. • If LOAD = 0, the timer stops at the end of the time interval.	0
31	LOAD		Determines how the timer interval value IVALUE -1 is loaded into the TIMERn register. This bit is write-only. Reading this bit always returns 0.	0
		0	No force load. The load from the INTVALn register to the TIMERn register is processed at the end of the time interval if the repeat mode is selected.	
		1	Force load. The INTVALn interval value IVALUE -1 is immediately loaded into the TIMERn register while TIMERn is running.	

11.6.2 Timer register

The timer register holds the current timer value. This register is read-only.

Table 137. Timer register (TIMER[0:3], address 0x4000 4004 (TIMER0) to 0x4000 4034 (TIMER3)) bit description

Bit	Symbol	Description	Reset value
30:0	VALUE	Holds the current timer value of the down-counter. The initial value of the TIMERn register is loaded as IVALUE - 1 from the INTVALn register either at the end of the time interval or immediately in the following cases: INTVALn register is updated in the idle state. INTVALn register is updated with LOAD = 1. When the timer is in idle state, reading this bit fields returns -1 (0x00FF FFFF).	0x00FF FFFF
31	-	Reserved.	0

11.6.3 Control register

The control register configures the the mode for each MRT and enables the interrupt.

Table 138. Control register (CTRL[0:3], address 0x4000 4008 (CTRL0) to 0x4000 4038 (CTRL3)) bit description

Bit	Symbol	Value	Description	Reset value
0	INTEN		Enable the TIMERn interrupt.	0
		0	Disable.	
		1	Enable.	
2:1	MODE		Selects timer mode.	0
		0x0	Repeat interrupt mode.	
		0x1	One-shot interrupt mode.	
		0x2	Reserved.	
		0x3	Reserved.	
31:3	-		Reserved.	0

11.6.4 Status register

This register indicates the status of each MRT.

Table 139. Status register (STAT[0:3], address 0x4000 400C (STAT0) to 0x4000 403C (STAT3)) bit description

Bit	Symbol	Value	Description	Reset value
0	INTFLAG		Monitors the interrupt flag.	0
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMERn has reached the end of the time interval. If the INTEN bit in the CONTROLn is also set to 1, the interrupt for timer channel n and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
1	RUN		Indicates the state of TIMERn. This bit is read-only.	0
		0	Idle state. TIMERn is stopped.	
		1	Running. TIMERn is running.	
31:2	-		Reserved.	0

11.6.5 Idle channel register

The idle channel register returns the lowest idle channel number. The channel is considered idle when both flags is the STATUS register (RUN and INTFLAG) are zero.

In an application with multiple timers running independently, you can calculate the register offset of the next idle timer by reading the idle channel number in this register. The idle channel register allows you set up the next idle timer without checking the idle state of each timer.

Table 140. Idle channel register (IDLE_CH, address 0x4000 40F4) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	0
7:4	CHAN	Idle channel. Reading the CHAN bits, returns the lowest idle timer channel. If all timer channels are running, CHAN = 4.	0
31:8	-	Reserved.	0

11.6.6 Global interrupt flag register

The global interrupt register combines the interrupt flags from the individual timer channels in one register. Setting and clearing each flag behaves in the same way as setting and clearing the INTFLAG bit in each of the STATUSn registers.

Table 141. Global interrupt flag register (IRQ_FLAG, address 0x4000 40F8) bit description

Bit	Symbol	Value	Description	Reset value
0	GFLAG0		Monitors the interrupt flag of TIMER0.	0
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMER0 has reached the end of the time interval. If the INTEN bit in the CONTROL0 register is also set to 1, the interrupt for timer channel 0 and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
1	GFLAG1		Monitors the interrupt flag of TIMER1.	0
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMER1 has reached the end of the time interval. If the INTEN bit in the CONTROL1 register is also set to 1, the interrupt for timer channel 1 and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
2	GFLAG2		Monitors the interrupt flag of TIMER2.	0
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMER2 has reached the end of the time interval. If the INTEN bit in the CONTROL2 register is also set to 1, the interrupt for timer channel 2 and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
3	GFLAG3		Monitors the interrupt flag of TIMER3.	0
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMER3 has reached the end of the time interval. If the INTEN bit in the CONTROL3 register is also set to 1, the interrupt for timer channel 3 and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
31:4	-		Reserved.	0

12.1 How to read this chapter

The watchdog timer is identical on all LPC800 parts.

12.2 Features

- Internally resets chip if not reloaded during the programmable time-out period.
- Optional windowed operation requires reload to occur between a minimum and maximum time-out period, both programmable.
- Optional warning interrupt can be generated at a programmable time prior to watchdog time-out.
- Programmable 24-bit timer with internal fixed pre-scaler.
- Selectable time period from 1,024 watchdog clocks ($T_{WDCLK} \times 256 \times 4$) to over 67 million watchdog clocks ($T_{WDCLK} \times 2^{24} \times 4$) in increments of 4 watchdog clocks.
- “Safe” watchdog operation. Once enabled, requires a hardware reset or a Watchdog reset to be disabled.
- Incorrect feed sequence causes immediate watchdog event if enabled.
- The watchdog reload value can optionally be protected such that it can only be changed after the “warning interrupt” time is reached.
- Flag to indicate Watchdog reset.
- The Watchdog clock (WDCLK) source is the WatchDog oscillator.
- The Watchdog timer can be configured to run in Deep-sleep or Power-down mode.
- Debug mode.

12.3 Basic configuration

The WWDT is configured through the following registers:

- Power to the register interface (WWDT PCLK clock): In the SYSAHBCLKCTRL register, set bit 17 in Table 18.
- Enable the WWDT clock source (the watchdog oscillator) in the PDRUNCFG register (Table 37). This is the clock source for the timer base.
- For waking up from a WWDT interrupt, enable the watchdog interrupt for wake-up in the STARTERP1 register (Table 34).

12.4 Pin description

The WWDT has no external pins.

12.5 General description

The purpose of the Watchdog Timer is to reset or interrupt the microcontroller within a programmable time if it enters an erroneous state. When enabled, a watchdog reset is generated if the user program fails to feed (reload) the Watchdog within a predetermined amount of time.

When a watchdog window is programmed, an early watchdog feed is also treated as a watchdog event. This allows preventing situations where a system failure may still feed the watchdog. For example, application code could be stuck in an interrupt service that contains a watchdog feed. Setting the window such that this would result in an early feed will generate a watchdog event, allowing for system recovery.

The Watchdog consists of a fixed (divide by 4) pre-scaler and a 24-bit counter which decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum Watchdog interval is ($T_{WDCLK} \times 256 \times 4$) and the maximum Watchdog interval is ($T_{WDCLK} \times 2^{24} \times 4$) in multiples of ($T_{WDCLK} \times 4$). The Watchdog should be used in the following manner:

- Set the Watchdog timer constant reload value in the TC register.
- Set the Watchdog timer operating mode in the MOD register.
- Set a value for the watchdog window time in the WINDOW register if windowed operation is desired.
- Set a value for the watchdog warning interrupt in the WARNINT register if a warning interrupt is desired.
- Enable the Watchdog by writing 0xAA followed by 0x55 to the FEED register.
- The Watchdog must be fed again before the Watchdog counter reaches zero in order to prevent a watchdog event. If a window value is programmed, the feed must also occur after the watchdog counter passes that value.

When the Watchdog Timer is configured so that a watchdog event will cause a reset and the counter reaches zero, the CPU will be reset, loading the stack pointer and program counter from the vector table as for an external reset. The Watchdog time-out flag (WDTOF) can be examined to determine if the Watchdog has caused the reset condition. The WDTOF flag must be cleared by software.

When the Watchdog Timer is configured to generate a warning interrupt, the interrupt will occur when the counter matches the value defined by the WARNINT register.

12.5.1 Block diagram

The block diagram of the Watchdog is shown below in the Figure 22. The synchronization logic (PCLK - WDCLK) is not shown in the block diagram.

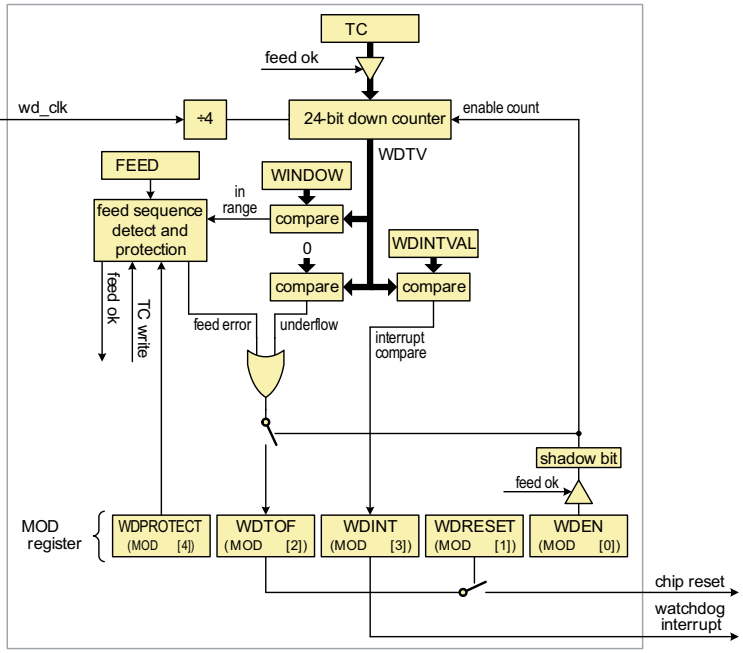


Fig 22. Windowed Watchdog timer block diagram

12.5.2 Clocking and power control

The watchdog timer block uses two clocks: PCLK and WDCLK. PCLK is used for the APB accesses to the watchdog registers and is derived from the system clock (see Figure 3). The WDCLK is used for the watchdog timer counting and is derived from the watchdog oscillator.

The synchronization logic between the two clock domains works as follows: When the MOD and TC registers are updated by APB operations, the new value will take effect in 3 WDCLK cycles on the logic in the WDCLK clock domain.

When the watchdog timer is counting on WDCLK, the synchronization logic will first lock the value of the counter on WDCLK and then synchronize it with PCLK, so that the CPU can read the WDTV register.

Remark: Because of the synchronization step, software must add a delay of three WDCLK clock cycles between the feed sequence and the time the WDPROTECT bit is enabled in the MOD register. The length of the delay depends on the selected watchdog clock WDCLK.

12.5.3 Using the WWDT lock features

The WWDT supports several lock features which can be enabled to ensure that the WWDT is running at all times:

- Disabling the WWDT clock source
- Changing the WWDT reload value

12.5.3.1 Disabling the WWDT clock source

If bit 5 in the WWDT MOD register is set, the WWDT clock source is locked and can not be disabled either by software or by hardware when Sleep, Deep-sleep or Power-down modes are entered. Therefore, the user must ensure that the watchdog oscillator for each power mode is enabled **before** setting bit 5 in the MOD register.

In Deep power-down mode, no clock locking mechanism is in effect because no clocks are running. However, an additional lock bit in the PMU can be set to prevent the part from even entering Deep power-down mode (see Table 43).

12.5.3.2 Changing the WWDT reload value

If bit 4 is set in the WWDT MOD register, the watchdog time-out value (TC) can be changed only after the counter is below the value of WDWARNINT and WDWINDOW.

The reload overwrite lock mechanism can only be disabled by a reset of any type.

12.6 Register description

The Watchdog Timer contains the registers shown in Table 142.

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

Table 142. Register overview: Watchdog timer (base address 0x4000 4000)

Name	Access	Address offset	Description	Reset value	Reference
MOD	R/W	0x000	Watchdog mode register. This register contains the basic mode and status of the Watchdog Timer.	0	Table 143
TC	R/W	0x004	Watchdog timer constant register. This 24-bit register determines the time-out value.	0xFF	Table 145
FEED	WO	0x008	Watchdog feed sequence register. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in WDTC.	NA	Table 146
TV	RO	0x00C	Watchdog timer value register. This 24-bit register reads out the current value of the Watchdog timer.	0xFF	Table 147
-	-	0x010	Reserved	-	-
WARNINT	R/W	0x014	Watchdog Warning Interrupt compare value.	0	Table 148
WINDOW	R/W	0x018	Watchdog Window compare value.	0xFF FFFF	Table 149

12.6.1 Watchdog mode register

The WDMOD register controls the operation of the Watchdog. Note that a watchdog feed must be performed before any changes to the WDMOD register take effect.

Table 143. Watchdog mode register (MOD, 0x4000 4000) bit description

Bit	Symbol	Value	Description	Reset value
0	WDEN		Watchdog enable bit. Once this bit has been written with a 1, it cannot be re-written with a 0. Once this bit is set to one, the watchdog timer starts running after a watchdog feed.	0
		0	The watchdog timer is stopped.	
		1	The watchdog timer is running.	
1	WDRESET		Watchdog reset enable bit. Once this bit has been written with a 1 it cannot be re-written with a 0.	0
		0	A watchdog time-out will not cause a chip reset.	
		1	A watchdog time-out will cause a chip reset.	
2	WDTOF		Watchdog time-out flag. Set when the watchdog timer times out, by a feed error, or by events associated with WDPROTECT. Cleared by software. Causes a chip reset if WDRESET = 1.	0 (only after external reset)

Table 143. Watchdog mode register (MOD, 0x4000 4000) bit description

Bit	Symbol	Value	Description	Reset value
3	WDINT		Warning interrupt flag. Set when the timer reaches the value in WDWARNINT. Cleared by software.	0
4	WDPROTECT		Watchdog update mode. This bit can be set once by software and is only cleared by a reset.	0
		0	The watchdog time-out value (TC) can be changed at any time.	
		1	The watchdog time-out value (TC) can be changed only after the counter is below the value of WDWARNINT and WDWINDOW.	
5	LOCK		A 1 in this bit prevents disabling or powering down the watchdog oscillator. This bit can be set once by software and is only cleared by any reset.	0
31:6	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Once the **WDEN**, **WDPROTECT**, or **WDRESET** bits are set they can not be cleared by software. Both flags are cleared by an external reset or a Watchdog timer reset.

WDTOF The Watchdog time-out flag is set when the Watchdog times out, when a feed error occurs, or when PROTECT =1 and an attempt is made to write to the TC register. This flag is cleared by software writing a 0 to this bit.

WDINT The Watchdog interrupt flag is set when the Watchdog counter reaches the value specified by WARNINT. This flag is cleared when any reset occurs, and is cleared by software by writing a 0 to this bit.

In all power modes except Deep power-down mode, a Watchdog reset or interrupt can occur when the watchdog is running and has an operating clock source. The watchdog oscillator can be configured to keep running in Sleep, Deep-sleep modes, and Power-down modes.

If a watchdog interrupt occurs in Sleep, Deep-sleep mode, or Power-down mode, and the WWDT interrupt is enabled in the NVIC, the device will wake up. Note that in Deep-sleep and Power-down modes, the WWDT interrupt must be enabled in the STARTERP1 register in addition to the NVIC.

See the following registers:

Table 34 “Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description”

Table 144. 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.

12.6.2 Watchdog Timer Constant register

The TC register determines the time-out value. Every time a feed sequence occurs the value in the TC is loaded into the Watchdog timer. The TC resets to 0x00 00FF. Writing a value below 0xFF will cause 0x00 00FF to be loaded into the TC. Thus the minimum time-out interval is $T_{WDCLK} \times 256 \times 4$.

If the WDPROTECT bit in WDMOD = 1, an attempt to change the value of TC before the watchdog counter is below the values of WDWARNINT and WDWINDOW will cause a watchdog reset and set the WDTOF flag.

Table 145. Watchdog Timer Constant register (TC, 0x4000 4004) bit description

Bit	Symbol	Description	Reset Value
23:0	COUNT	Watchdog time-out value.	0x00 00FF
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.6.3 Watchdog Feed register

Writing 0xAA followed by 0x55 to this register will reload the Watchdog timer with the WDTC value. This operation will also start the Watchdog if it is enabled via the WDMOD register. Setting the WDEN bit in the WDMOD register is not sufficient to enable the Watchdog. A valid feed sequence must be completed after setting WDEN before the Watchdog is capable of generating a reset. Until then, the Watchdog will ignore feed errors.

After writing 0xAA to WDFEED, access to any Watchdog register other than writing 0x55 to WDFEED causes an immediate reset/interrupt when the Watchdog is enabled, and sets the WDTOF flag. The reset will be generated during the second PCLK following an incorrect access to a Watchdog register during a feed sequence.

It is good practice to disable interrupts around a feed sequence, if the application is such that an interrupt might result in rescheduling processor control away from the current task in the middle of the feed, and then lead to some other access to the WDT before control is returned to the interrupted task.

Table 146. Watchdog Feed register (FEED, 0x4000 4008) bit description

Bit	Symbol	Description	Reset Value
7:0	FEED	Feed value should be 0xAA followed by 0x55.	NA
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.6.4 Watchdog Timer Value register

The WDTV register is used to read the current value of Watchdog timer counter.

When reading the value of the 24-bit counter, the lock and synchronization procedure takes up to 6 WDCLK cycles plus 6 PCLK cycles, so the value of WDTV is older than the actual value of the timer when it's being read by the CPU.

Table 147. Watchdog Timer Value register (TV, 0x4000 400C) bit description

Bit	Symbol	Description	Reset Value
23:0	COUNT	Counter timer value.	0x00 00FF
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.6.5 Watchdog Timer Warning Interrupt register

The WDWARNINT register determines the watchdog timer counter value that will generate a watchdog interrupt. When the watchdog timer counter matches the value defined by WARNINT, an interrupt will be generated after the subsequent WDCLK.

A match of the watchdog timer counter to WARNINT occurs when the bottom 10 bits of the counter have the same value as the 10 bits of WARNINT, and the remaining upper bits of the counter are all 0. This gives a maximum time of 1,023 watchdog timer counts (4,096 watchdog clocks) for the interrupt to occur prior to a watchdog event. If WARNINT is 0, the interrupt will occur at the same time as the watchdog event.

Table 148. Watchdog Timer Warning Interrupt register (WARNINT, 0x4000 4014) bit description

Bit	Symbol	Description	Reset Value
9:0	WARNINT	Watchdog warning interrupt compare value.	0
31:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.6.6 Watchdog Timer Window register

The WINDOW register determines the highest WDTV value allowed when a watchdog feed is performed. If a feed sequence occurs when WDTV is greater than the value in WINDOW, a watchdog event will occur.

WINDOW resets to the maximum possible WDTV value, so windowing is not in effect.

Table 149. Watchdog Timer Window register (WINDOW, 0x4000 4018) bit description

Bit	Symbol	Description	Reset Value
23:0	WINDOW	Watchdog window value.	0xFF FFFF
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.7 Functional description

The following figures illustrate several aspects of Watchdog Timer operation.

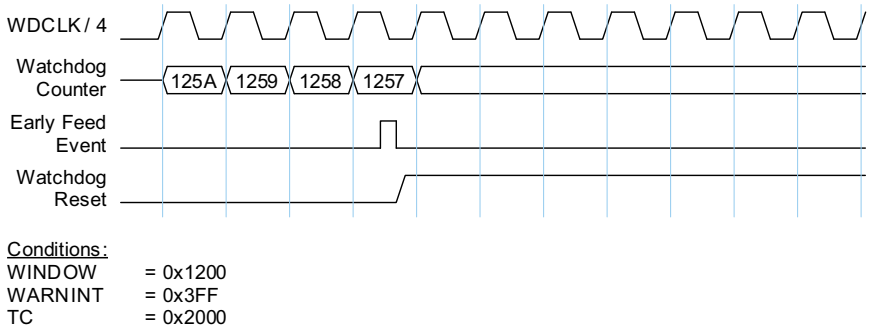


Fig 23. Early watchdog feed with windowed mode enabled

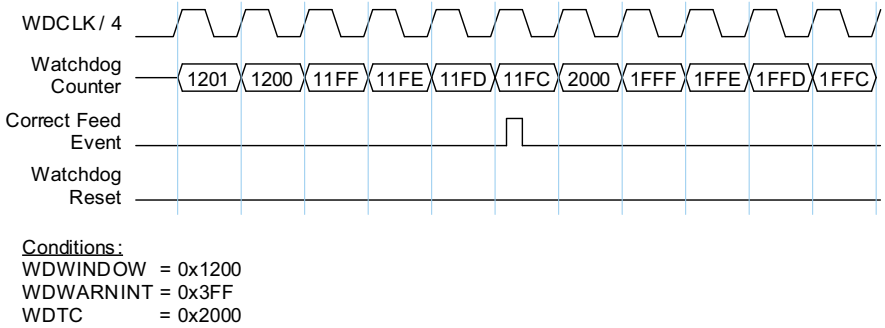


Fig 24. Correct watchdog feed with windowed mode enabled

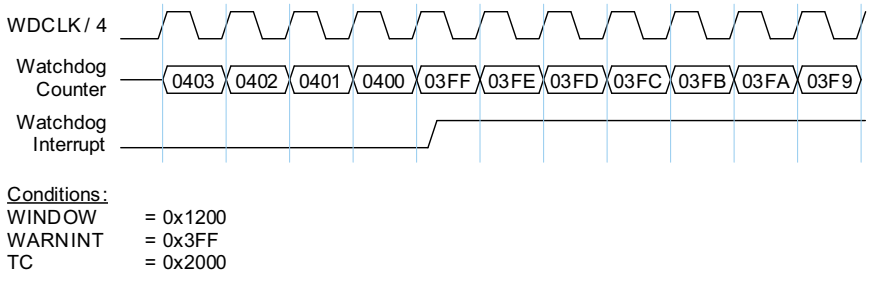


Fig 25. Watchdog warning interrupt

13.1 How to read this chapter

The self wake-up timer is available on all LPC800 parts.

13.2 Features

- 32-bit loadable down-counter. Counter starts automatically when a count value is loaded. Time-out generates an interrupt/wake up request.
- The WKT resides in a separate, always-on power domain.
- The WKT supports two clock sources. One clock source originates from the always-on power domain.
- The WKT can be used for waking up the part from any low power mode, including Deep power-down mode, or for general-purpose timing.

13.3 Basic configuration

- In the SYSAHBCLKCTRL register, set bit 9 (Table 18) to enable the clock to the register interface.
- Clear the WKT reset using the PRESETCTRL register (Table 7).
- The WKT interrupt is connected to interrupt #15 in the NVIC.
- Enable the low power oscillator in the PMU (Table 46).
- Enable the IRC and IRC output in the PDRUNCFG register (Table 37).
- See Section 5.7.1 to enable the various power down modes.

13.4 Pin description

The WKT has no configurable pins.

13.5 General description

The self wake-up timer is a 32-bit, loadable down-counter. Writing any non-zero value to this timer automatically enables the counter and launches a count-down sequence. When the counter is being used as a wake up timer, this write can occur just prior to entering a reduced power mode.

When a starting count value is loaded, the self wake-up timer automatically turns on, counts from the pre-loaded value down to zero, generates an interrupt and/or a wake up request, and then turns itself off until re-launched by a subsequent software write.

13.5.1 WKT clock sources

The self wake-up timer can be clocked from two alternative clock sources:

- A 750 kHz clock derived from the IRC oscillator. This is the default clock,

- A 10 kHz, low-power clock with a dedicated on-chip oscillator as clock source.

The IRC-derived clock is much more accurate than the alternative, low-power clock. However, the IRC is not available in most low-power modes. This clock must not be selected when the timer is being used to wake up from a power mode where the IRC is disabled.

The alternative clock source is a (nominally) 10 kHz, low-power clock, sourced from a dedicated oscillator. This oscillator resides in the always-on voltage domain, so it can be programmed to continue operating in Deep power-down mode when power is removed from the rest of the part. This clock is also be available during other low-power modes when the IRC clock is shut-down.

The Low-Power oscillator is not accurate (approximately +/- 40 % over process and temperature). The frequency may still drift while counting is in progress due to reduced chip temperature after a low-power mode is entered.

13.6 Register description

Table 150. Register overview: WKT (base address 0x4000 8000)

Name	Access	Address offset	Description	Reset value	Reference
CTRL	R/W	0x0	Self wake-up timer control register.	0	Table 151
COUNT	R/W	0xC	Counter register.		

13.6.1 Control register

The WKT interrupt must be enabled in the NVIC to wake up the part using the self wake-up counter.

Table 151. Control register (CTRL, address 0x4000 8000) bit description

Bit	Symbol	Value	Description	Reset value
0	CLKSEL		Select the self wake-up timer clock source.	0
		0	Divided IRC clock. This clock runs at 750 kHz and provides time-out periods of up to approximately 95 minutes in 1.33 µs increments. **Remark:** This clock is not available in not available in Deep-sleep, power-down, deep power-down modes. Do not select this option if the timer is to be used to wake up from one of these modes.	
		1	Low power clock. This is the (nominally) 10 kHz clock and provides time-out periods of up to approximately 119 hours in 100 µs increments. The accuracy of this clock is limited to +/- 45 % over temperature and processing. **Remark:** This clock is available in all power modes. Prior to use, the low-power oscillator must be enabled. The oscillator must also be set to remain active in Deep power-down if needed.	
1	ALARMFLAG		Wake-up or alarm timer flag.	-
		0	No time-out. The self wake-up timer has not timed out. Writing a 0 to has no effect.	
		1	Time-out. The self wake-up timer has timed out. This flag generates an interrupt request which can wake up the part from any reduced power mode including Deep power-down if the clock source is the low power oscillator. Writing a 1 clears this status bit.	

Table 151. Control register (CTRL, address 0x4000 8000) bit description

Bit	Symbol	Value	Description	Reset value
2	CLEARCTR		Clears the self wake-up timer.	0
		0	No effect. Reading this bit always returns 0.	
		1	Clear the counter. Counting is halted until a new count value is loaded.	
31:3	-		Reserved.	-

13.6.2 Count register

Do not write to this register while the counting is in progress.

Remark: In general, reading the timer state is not recommended. There is no mechanism to ensure that some bits of this register don't change while a read is in progress if the read happens to coincide with an self wake-up timer clock edge. If you must read this value, it is recommended to read it twice in succession.

Table 152. Counter register (COUNT, address 0x4000 800C) bit description

Bit	Symbol	Description	Reset value
31:0	VALUE	A write to this register pre-loads start count value into the timer and starts the count-down sequence. A read reflects the current value of the timer.	-

14.1 How to read this chapter

The SysTick timer is available on all LPC800 parts.

14.2 Features

- Simple 24-bit timer.
- Uses dedicated exception vector.
- Clocked internally by the system clock or the system clock/2.

14.3 Basic configuration

The system tick timer is configured using the following registers:

1. The system tick timer is enabled through the SysTick control register (Table 154). The system tick timer clock is fixed to half the frequency of the system clock.
2. Enable the clock source for the SysTick timer in the SYST_CSR register (Table 154).

14.4 Pin description

TheSysTick has no configurable pins.

14.5 General description

The block diagram of the SysTick timer is shown below in the Figure 26.

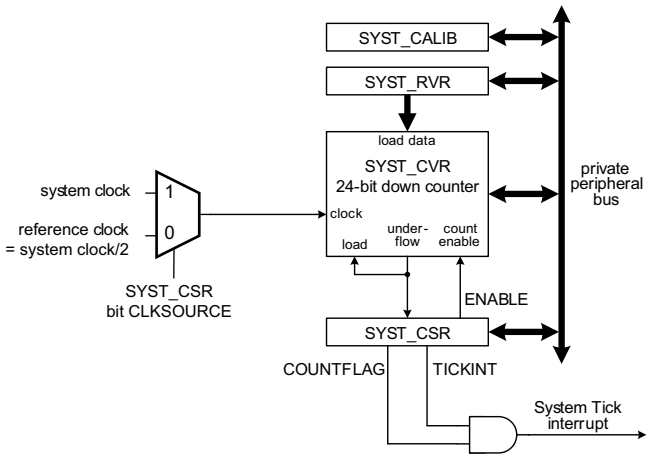


Fig 26. System tick timer block diagram

The SysTick timer is an integral part of the Cortex-M0+. The SysTick timer is intended to generate a fixed 10 millisecond interrupt for use by an operating system or other system management software.

Since the SysTick timer is a part of the Cortex-M0+, it facilitates porting of software by providing a standard timer that is available on Cortex-M0 based devices. The SysTick timer can be used for:

- An RTOS tick timer which fires at a programmable rate (for example 100 Hz) and invokes a SysTick routine.
- A high-speed alarm timer using the core clock.
- A simple counter. Software can use this to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNTFLAG bit-field in the control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

Refer to Ref. 3 for details.

14.6 Register description

The SysTick timer registers are located on the ARM Cortex-M0+ private peripheral bus (see Figure 2), and are part of the ARM Cortex-M0+ core peripherals. For details, see Ref. 3.

Table 153. Register overview: SysTick timer (base address 0xE000 E000)

Name	Access	Address offset	Description	Reset value[1]
SYST_CSR	R/W	0x010	System Timer Control and status register	0x000 0000
SYST_RVR	R/W	0x014	System Timer Reload value register	0
SYST_CVR	R/W	0x018	System Timer Current value register	0
SYST_CALIB	R/W	0x01C	System Timer Calibration value register	0x4

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

14.6.1 System Timer Control and status register

The SYST_CSR register contains control information for the SysTick timer and provides a status flag. This register is part of the ARM Cortex-M0+ core system timer register block. For a bit description of this register, see Ref. 3.

This register determines the clock source for the system tick timer.

Table 154. SysTick Timer Control and status register (SYST_CSR - 0xE000 E010) bit description

Bit	Symbol	Description	Reset value
0	ENABLE	System Tick counter enable. When 1, the counter is enabled. When 0, the counter is disabled.	0
1	TICKINT	System Tick interrupt enable. When 1, the System Tick interrupt is enabled. When 0, the System Tick interrupt is disabled. When enabled, the interrupt is generated when the System Tick counter counts down to 0.	0
2	CLKSOURCE	System Tick clock source selection. When 1, the system clock (CPU) clock is selected. When 0, the system clock/2 is selected as the reference clock.	0
15:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
16	COUNTFLAG	Returns 1 if the SysTick timer counted to 0 since the last read of this register.	0
31:17	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.6.2 System Timer Reload value register

The SYST_RVR register is set to the value that will be loaded into the SysTick timer whenever it counts down to zero. This register is loaded by software as part of timer initialization. The SYST_CALIB register may be read and used as the value for SYST_RVR register if the CPU is running at the frequency intended for use with the SYST_CALIB value.

Table 155. System Timer Reload value register (SYST_RVR - 0xE000 E014) bit description

Bit	Symbol	Description	Reset value
23:0	RELOAD	This is the value that is loaded into the System Tick counter when it counts down to 0.	0
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.6.3 System Timer Current value register

The SYST_CVR register returns the current count from the System Tick counter when it is read by software.

Table 156. System Timer Current value register (SYST_CVR - 0xE000 E018) bit description

Bit	Symbol	Description	Reset value
23:0	CURRENT	Reading this register returns the current value of the System Tick counter. Writing any value clears the System Tick counter and the COUNTFLAG bit in STCTRL.	0
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.6.4 System Timer Calibration value register (SYST_CALIB - 0xE000 E01C)

The value of the SYST_CALIB register is driven by the value of the SYSTCKCAL register in the system configuration block SYSCON (see Table 29).

Table 157. System Timer Calibration value register (SYST_CALIB - 0xE000 E01C) bit description

Bit	Symbol	Value	Description	Reset value
23:0	TENMS		See Ref. 3.	0x4
29:24	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
30	SKEW		See Ref. 3.	0
31	NOREF		See Ref. 3.	0

14.7 Functional description

The SysTick timer is a 24-bit timer that counts down to zero and generates an interrupt. The intent is to provide a fixed 10 millisecond time interval between interrupts. The SysTick timer is clocked from the CPU clock (the system clock, see Figure 3) or from the reference clock, which is fixed to half the frequency of the CPU clock. In order to generate recurring interrupts at a specific interval, the SYST_RVR register must be initialized with the correct value for the desired interval. A default value is provided in the SYST_CALIB register and may be changed by software. The default value gives a 10 millisecond interrupt rate if the CPU clock is set to <tbd> MHz.

14.7.1 Example timer calculation

To use the system tick timer, do the following:

1. Program the SYST_RVR register with the reload value RELOAD to obtain the desired time interval.
2. Clear the SYST_CVR register by writing to it. This ensures that the timer will count from the SYST_RVR value rather than an arbitrary value when the timer is enabled.
3. Program the SYST_SCR register with the value 0x7 which enables the SysTick timer and the SysTick timer interrupt.

The following example illustrates selecting the SysTick timer reload value to obtain a 10 ms time interval with the system clock set to 20 MHz.

Example (system clock = 20 MHz)

The system tick clock = system clock = 20 MHz. Bit CLKSOURCE in the SYST_CSR register set to 1 (system clock).

RELOAD = (system tick clock frequency $\times$ 10 ms) -1 = (20 MHz $\times$ 10 ms) -1 = 200000 -1 = 199999 = 0x00030D3F.

15.1 How to read this chapter

USART0 and USART1 are available on all parts. USART2 is available on parts LPC812M101FDH16 and LPC812M101FDH20 only.

Read this chapter for a description of the USART peripheral and the software interface.

The LPC800 also provides an on-chip ROM-based USART API to configure and operate the USART. See Table 279.

15.2 Features

- 7, 8, or 9 data bits and 1 or 2 stop bits
- Synchronous mode with master or slave operation. Includes data phase selection and continuous clock option.
- Multiprocessor/multidrop (9-bit) mode with software address compare. (RS-485 possible with software address detection and transceiver direction control.)
- Parity generation and checking: odd, even, or none.
- One transmit and one receive data buffer.
- RTS/CTS for hardware signaling for automatic flow control. Software flow control can be performed using Delta CTS detect, Transmit Disable control, and any GPIO as an RTS output.
- Received data and status can optionally be read from a single register
- Break generation and detection.
- Receive data is 2 of 3 sample "voting". Status flag set when one sample differs.
- Built-in Baud Rate Generator.
- A fractional rate divider is shared among all USARTs.
- Interrupts available for Receiver Ready, Transmitter Ready, Receiver Idle, change in receiver break detect, Framing error, Parity error, Overrun, Underrun, Delta CTS detect, and receiver sample noise detected.
- Loopback mode for testing of data and flow control.

15.3 Basic configuration

Remark: The on-chip USART API provides software routines to configure and use the USART. See Table 279.

Configure USART0/1/2 for receiving and transmitting data:

- In the SYSAHBCLKCTRL register, set bit 14 to 16 (Table 18) to enable the clock to the register interface.
- Clear the USART0/1/2 peripheral resets using the PRESETCTRL register (Table 7).
- Enable or disable the USART0/1/2 interrupts in slots #3 to 5 in the NVIC.

- Configure the USART0/1/2 pin functions through the switch matrix. See Section 15.4.
- Configure the USART clock and baud rate. See Section 15.3.1.

Configure the USART0/1/2 to wake up the part from low power modes:

- Configure the USART to receive and transmit data in synchronous slave mode. See Section 15.3.2.

15.3.1 Configure the USART clock and baud rate

All three USARTs use a common peripheral clock (U_PCLK) and, if needed, a fractional baud rate generator. The peripheral clock and the fractional divider for the baud rate calculation are set up in the SYSCON block as follows (see Figure 27):

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

   Section 4.6.14 “USART clock divider register”
2. If a fractional value is needed to obtain a particular baud rate, program the fractional divider. The fractional divider value is the fraction of MULT/DIV. The MULT value is programmed in the UARTFRGMULT register and the DIV value is programmedwith the fixed value of 256 in the UARTFRGDIV register in the SYSCON block.

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

   The following rules apply for MULT and DIV:

   – Always set DIV to 256 by programming the UARTFRGDIV register with the value of 0xFF.

   – Program any value between 0 and 255 in the UARTFRGMULT register.

   Section 4.6.19 “USART fractional generator multiplier value register”

   Section 4.6.18 “USART fractional generator divider value register”
3. In asynchronous mode: Configure the baud rate divider BRGVAL in the USARTn BRG register. The baud rate divider divides the common USART peripheral clock by a factor of 16 multiplied by the baud rate value to provide the baud rate = U_PCLK/16 x BRGVAL.

   Section 15.6.9 “USART Baud Rate Generator register”
4. In synchronous mode: The serial clock is Un_SCLK = U_PCLK/BRGVAL.

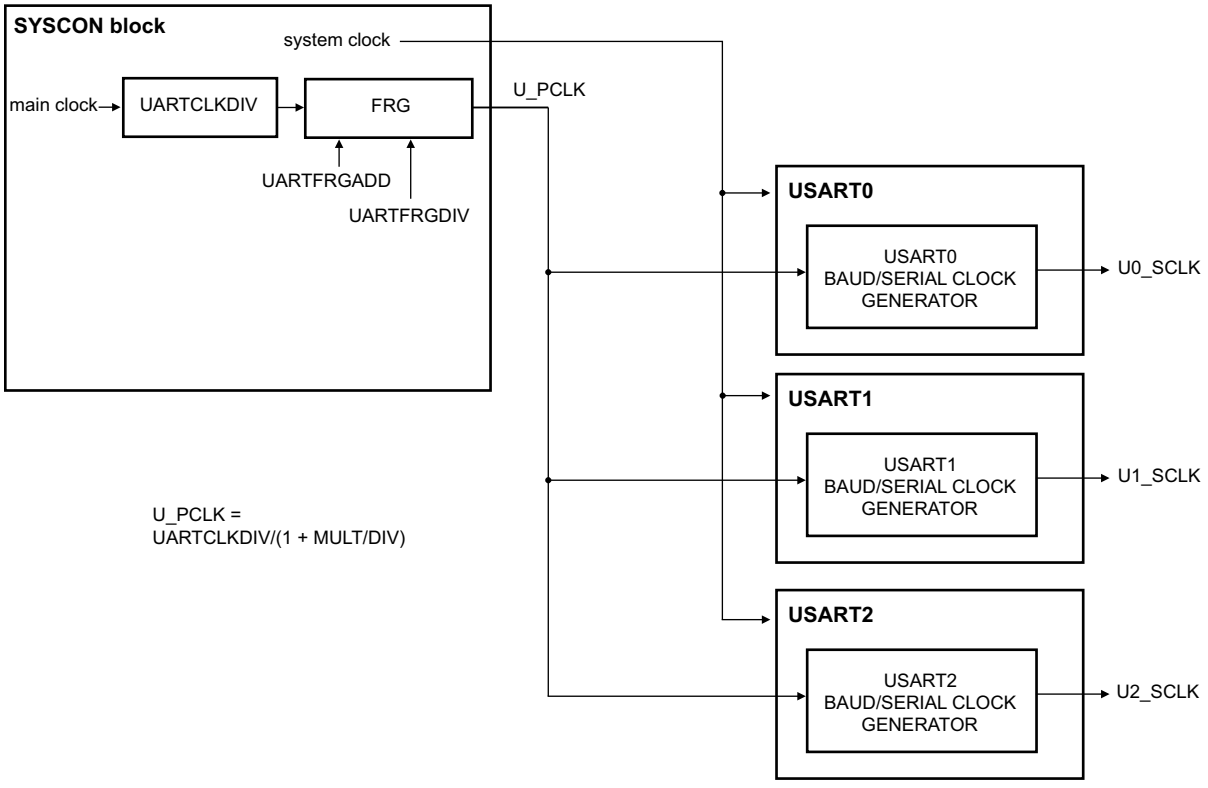


Fig 27. USART clocking

For details on the clock configuration see:

Section 15.7.1 “Clocking and Baud rates”

15.3.2 Configure the USART for wake-up

The USART can wake up the system from sleep mode in asynchronous or synchronous mode on any enabled USART interrupt.

If the USART is configured for synchronous slave mode, the USART block can create an interrupt on a received signal even when the USART block receives no clocks from the ARM Cortex-M0+ core - that is in Deep-sleep or Power-down mode.

As long as the USART receives a clock signal from the master, it can receive up to one byte in the RXDATA register while in Deep-sleep or Power-down mode. Any interrupt raised as part of the receive data process can then wake up the part.

15.3.2.1 Wake-up from Sleep mode

- Configure the USART in either asynchronous mode or synchronous mode. See Table 160.
- Enable the USART interrupt in the NVIC.
- Any USART interrupt wakes up the part from sleep mode. Enable the USART interrupt in the INTENSET register (Table 163).

15.3.2.2 Wake-up from Deep-sleep or Power-down mode

- Configure the USART in synchronous slave mode. See Table 160. You must connect the SCLK function to a pin and connect the pin to the master.
- Enable the USART interrupt in the STARTERP1 register. See Table 34 “Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description”.
- Enable the USART interrupt in the NVIC.
- In the PDAWAKE register, configure all peripherals that need to be running when the part wakes up.
- The USART wakes up the part from Deep-sleep or Power-down mode on all events that cause an interrupt and areal so enabled in the INTENSET register. Typical wake-up events are:
  - A start bit has been received.
  - The RXDATA buffer has received a byte.
  - Data is ready to be transmitted in the TXDATA buffer and a serial clock from the master has been received.
  - A change in the state of the CTS pin if the CTS function is connected. <tbd>
  - **Remark:** By enabling or disabling the interrupt in the INTENSET register (Table 163), you can customize when the wake-up occurs in the USART receive/transmit protocol.

15.4 Pin description

The USART receive, transmit, and control signals are movable functions and are assigned to external pins through the switch matrix.

See Section 9.3.1 “Connect an internal signal to a package pin” to assign the USART functions to pins on the LPC800 package.

Table 158. USART pin description

Function	Direction	Pin	Description	SWM register	Reference
U0_TXD	O	any	Transmitter output for USART0. Serial transmit data.	PINASSIGN0	Table 97
U0_RXD	I	any	Receiver input for USART0. Serial receive data.	PINASSIGN0	Table 97
$\overline{\text{U0\_RTS}}$	O	any	Request To Send output for USART0. Active low signal indicates that the USART0 is ready to receive data. This signal supports inter-processor communication through the use of hardware flow control. This feature is active when the USART RTS signal is configured to appear on a device pin.	PINASSIGN0	Table 97
$\overline{\text{U0\_CTS}}$	I	any	Clear To Send input for USART0. Active low signal indicates that the external device that is in communication with the USART is ready to accept data. This feature is active when enabled by the CTSEn bit in CFG register and when configured to appear on a device pin. When deasserted (high) by the external device, the USART will complete transmitting any character already in progress, then stop until CTS is again asserted (low).	PINASSIGN0	Table 97

Table 158. USART pin description

Function	Direction	Pin	Description	SWM register	Reference
U0_SCLK	I/O	any	Serial clock input/output for USART0 in synchronous mode. Clock input or output in synchronous mode.	PINASSIGN1	Table 98
U1_TXD	O	any	Transmitter output for USART1. Serial transmit data.	PINASSIGN1	Table 98
U1_RXD	I	any	Receiver input for USART1.	PINASSIGN1	Table 98
$\overline{\text{U1\_RTS}}$	O	any	Request To Send output for USART1.	PINASSIGN1	Table 98
$\overline{\text{U1\_CTS}}$	I	any	Clear To Send input for USART1.	PINASSIGN2	Table 99
U1_SCLK	I/O	any	Serial clock input/output for USART1 in synchronous mode.	PINASSIGN2	Table 99
U2_TXD	O	any	Transmitter output for USART2. Serial transmit data.	PINASSIGN2	Table 99
U2_RXD	I	any	Receiver input for USART2.	PINASSIGN2	Table 99
$\overline{\text{U2\_RTS}}$	O	any	Request To Send output for USART2.	PINASSIGN3	Table 100
$\overline{\text{U2\_CTS}}$	I	any	Clear To Send input for USART2.	PINASSIGN3	Table 100
U2_SCLK	I/O	any	Serial clock input/output for USART2 in synchronous mode.	PINASSIGN3	Table 100

15.5 General description

The USART receiver block monitors the serial input line, Un_RXD, for valid input. The receiver shift register assembles characters as they are received, after which they are passed to the receiver buffer register to await access by the CPU.

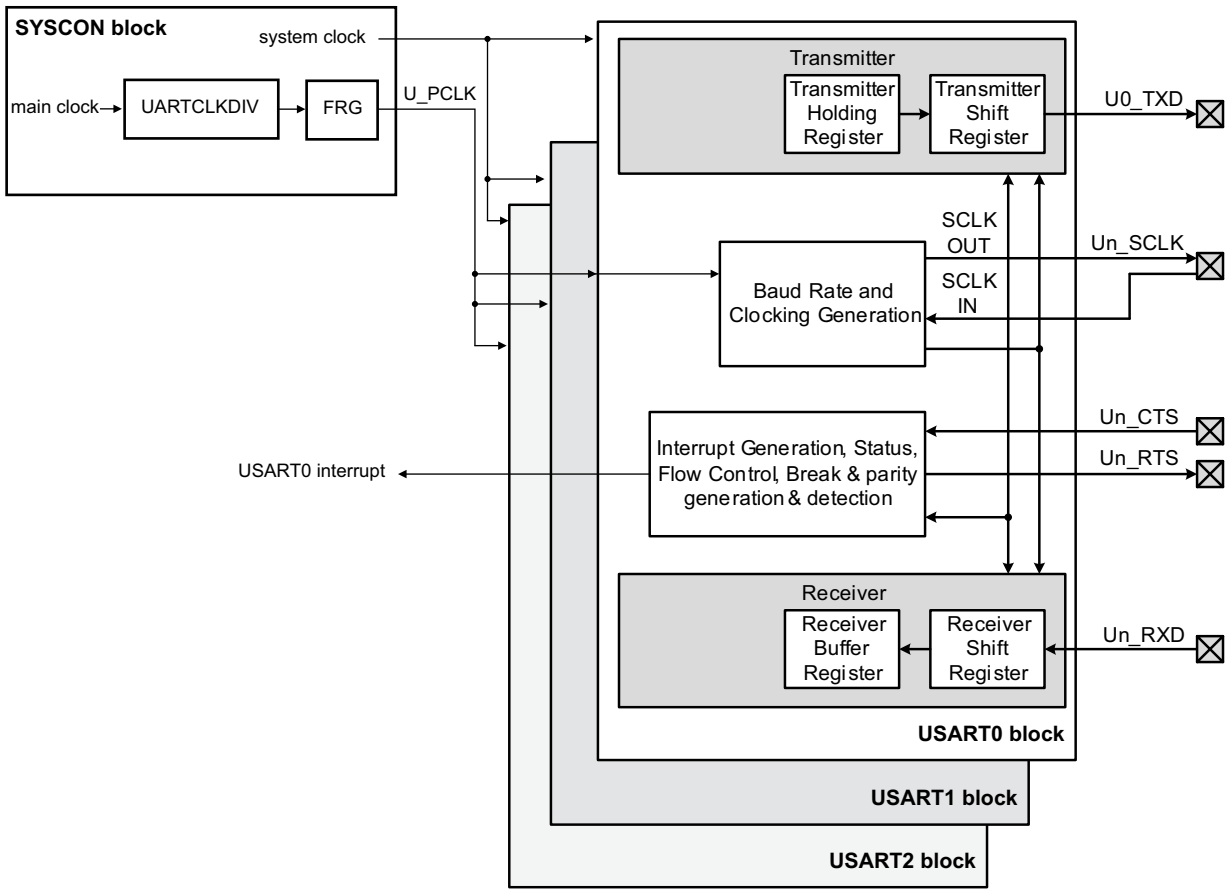
The USART transmitter block accepts data written by the CPU and buffers the data in the transmit holding register. When the transmitter is available, the transmit shift register takes that data, formats it, and serializes it to the serial output, Un_TXD.

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

In synchronous slave mode, data is transmitted and received using the serial clock directly. In synchronous master mode, data is transmitted and received using the baud rate clock without division.

Status information from the transmitter and receiver is saved and provided via the Stat register. Many of the status flags are able to generate interrupts, as selected by software.

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



U_PCLK = UARTCLKDIV/(1+MULT/DIV)

Fig 28. USART block diagram

15.6 Register description

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

Table 159: Register overview: USART (base address 0x4006 4000 (USART0), 0x4006 8000 (USART1), 0x4006 C000 (USART2))

Name	Access	Offset	Description	Reset value	Reference
CFG	R/W	0x000	USART Configuration register. Basic USART configuration settings that typically are not changed during operation.	0	Table 160
CTRL	R/W	0x004	USART Control register. USART control settings that are more likely to change during operation.	0	Table 161
STAT	R/W	0x008	USART Status register. The complete status value can be read here. Writing 1s clears some bits in the register. Some bits can be cleared by writing a 1 to them.	0x000E	Table 162
INTENSET	R/W	0x00C	Interrupt Enable read and Set register. Contains an individual interrupt enable bit for each potential USART interrupt. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set.	0	Table 163
INTENCLR	W	0x010	Interrupt Enable Clear register. Allows clearing any combination of bits in the INTENSET register. Writing a 1 to any implemented bit position causes the corresponding bit to be cleared.	-	Table 164
RXDATA	R	0x014	Receiver Data register. Contains the last character received.	-	Table 165
RXDATASTAT	R	0x018	Receiver Data with Status register. Combines the last character received with the current USART receive status. Allows software to recover incoming data and status together.	-	Table 166
TXDATA	R/W	0x01C	Transmit Data register. Data to be transmitted is written here.	0	Table 167
BRG	R/W	0x020	Baud Rate Generator register. 16-bit integer baud rate divisor value.	0	Table 168
INTSTAT	R	0x024	Interrupt status register. Reflects interrupts that are currently enabled.	0x0005	Table 169

15.6.1 USART Configuration register

The CFG register contains communication and mode settings for aspects of the USART that would normally be configured once in an application.

Remark: If software needs to change configuration values, the following sequence should be used: 1) Make sure the USART is not currently sending or receiving data. 2) Disable the USART by writing a 0 to the Enable bit (0 may be written to the entire register). 3) Write the new configuration value, with the ENABLE bit set to 1.

Table 160. USART Configuration register (CFG, address 0x4006 4000 (USART0), 0x4006 8000 (USART1), 0x4006 C000 (USART2)) bit description

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

Table 160. USART Configuration register (CFG, address 0x4006 4000 (USART0), 0x4006 8000 (USART1), 0x4006 C000 (USART2)) bit description *…continued*

Bit	Symbol	Value	Description	Reset Value
9	CTSEN		CTS Enable. Determines whether CTS is used for flow control. CTS can be from the input pin, or from the USART's own RTS if loopback mode is enabled. See Section 15.7.3 for more information.	0
		0	No flow control. The transmitter does not receive any automatic flow control signal.	
		1	Flow control enabled. The transmitter uses the CTS input (or RTS output in loopback mode) for flow control purposes.	
10	-		Reserved. Read value is undefined, only zero should be written.	NA
11	SYNCEN		Selects synchronous or asynchronous operation.	0
		0	Asynchronous mode is selected.	
		1	Synchronous mode is selected.	
12	CLKPOL		Selects the clock polarity and sampling edge of received data in synchronous mode.	0
		0	Falling edge. Un_RXD is sampled on the falling edge of SCLK.	
		1	Rising edge. Un_RXD is sampled on the rising edge of SCLK.	
13	-		Reserved. Read value is undefined, only zero should be written.	NA
14	SYNCMST		Synchronous mode Master select.	0
		0	Slave. When synchronous mode is enabled, the USART is a slave.	
		1	Master. When synchronous mode is enabled, the USART is a master.	
15	LOOP		Selects data loopback mode.	0
		0	Normal operation.	
		1	Loopback mode. This provides a mechanism to perform diagnostic loopback testing for USART data. Serial data from the transmitter (Un_TXD) is connected internally to serial input of the receive (Un_RXD). Un_TXD and Un_RTS activity will also appear on external pins if these functions are configured to appear on device pins. The receiver RTS signal is also looped back to CTS and performs flow control if enabled by CTSEN.	
31:16	-		Reserved. Read value is undefined, only zero should be written.	NA

15.6.2 USART Control register

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

Table 161. USART Control register (CTRL, address 0x4006 4004 (USART0), 0x4006 8004 (USART1), 0x4006 C004 (USART2)) bit description

Bit	Symbol	Value	Description	Reset Value
0	-		Reserved. Read value is undefined, only zero should be written.	NA
1	TXBRKEN		Break Enable.	0
		0	Normal operation.	
		1	Continuous break is sent immediately when this bit is set, and remains until this bit is cleared. A break may be sent without danger of corrupting any currently transmitting character if the transmitter is first disabled (TXDIS in CTRL is set) and then waiting for the transmitter to be disabled (TXDISINT in STAT = 1) before writing 1 to TXBRKEN.	
2	ADDRDET		Enable address detect mode.	0
		0	Enabled. The USART receiver is enabled for all incoming data.	
		1	Disabled. The USART receiver ignores incoming data that does not have the most significant bit of the data (typically the 9th bit) = 1. When the data MSB bit = 1, the receiver treats the incoming data normally, generating a received data interrupt. Software can then check the data to see if this is an address that should be handled. If it is, the ADDRDET bit is cleared by software and further incoming data is handled normally.	
5:3	-		Reserved. Read value is undefined, only zero should be written.	NA
6	TXDIS		Transmit Disable.	0
		0	Not disabled. USART transmitter is not disabled.	
		1	Disabled. USART transmitter is disabled after any character currently being transmitted is complete. This feature can be used to facilitate software flow control.	
7	-		Reserved. Read value is undefined, only zero should be written.	NA
8	CC		Continuous Clock generation. By default, SCLK is only output while data is being transmitted in synchronous mode.	0
		0	Clock on character. In synchronous mode, SCLK cycles only when characters are being sent on Un_TXD or to complete a character that is being received.	
		1	Continuous clock. SCLK runs continuously in synchronous mode, allowing characters to be received on Un_RxD independently from transmission on Un_TXD).	
9	CLRCC		Clear Continuous Clock.	0
		0	No affect on the CC bit.	
		1	Auto-clear. The CC bit is automatically cleared when a complete character has been received. This bit is cleared at the same time.	
31:10	-		Reserved. Read value is undefined, only zero should be written.	NA

15.6.3 USART Status register

The STAT register primarily provides a complete set of USART status flags for software to read. Flags other than read-only flags may be cleared by writing ones to corresponding bits of STAT. Interrupt status flags that are read-only and cannot be cleared by software, can be masked using the INTENCLR register (see Table 164).

The error flags for received noise, parity error, framing error, and overrun are set immediately upon detection and remain set until cleared by software action in STAT.

Table 162. USART Status register (STAT, address 0x4006 4008 (USART0), 0x4006 8008 (USART1), 0x4006 C008 (USART2)) bit description

Bit	Symbol	Description	Reset value	Access [1]
0	RXRDY	Receiver Ready flag. When 1, indicates that data is available to be read from the receiver buffer. Cleared after a read of the RXDATA or RXDATASTAT registers.	0	RO
1	RXIDLE	Receiver Idle. When 0, indicates that the receiver is currently in the process of receiving data. When 1, indicates that the receiver is not currently in the process of receiving data.	1	RO
2	TXRDY	Transmitter Ready flag. When 1, this bit indicates that data may be written to the transmit buffer. Previous data may still be in the process of being transmitted. Cleared when data is written to TXDATA. Set when the data is moved from the transmit buffer to the transmit shift register.	1	RO
3	TXIDLE	Transmitter Idle. When 0, indicates that the transmitter is currently in the process of sending data.When 1, indicate that the transmitter is not currently in the process of sending data.	1	RO
4	CTS	This bit reflects the current state of the CTS signal, regardless of the setting of the CTSEN bit in the CFG register. This will be the value of the CTS input pin unless loopback mode is enabled.	NA	RO
5	DELTACTS	This bit is set when a change in the state is detected for the CTS flag above. This bit is cleared by software.	0	W1
6	TXDISINT	Transmitter Disabled Interrupt flag. When 1, this bit indicates that the USART transmitter is fully idle after being disabled via the TXDIS in the CFG register (TXDIS = 1).	0	RO
7	-	Reserved. Read value is undefined, only zero should be written.	NA	NA
8	OVERRUNINT	Overrun Error interrupt flag. This flag is set when a new character is received while the receiver buffer is still in use. If this occurs, the newly received character in the shift register is lost.	0	W1
9	-	Reserved. Read value is undefined, only zero should be written.	NA	NA
10	RXBRK	Received Break. This bit reflects the current state of the receiver break detection logic. It is set when the Un_RXD pin remains low for 16 bit times. Note that FRAMERRINT will also be set when this condition occurs because the stop bit(s) for the character would be missing. RXBRK is cleared when the Un_RXD pin goes high.	0	RO
11	DELTARXBRK	This bit is set when a change in the state of receiver break detection occurs. Cleared by software.	0	W1
12	START	This bit is set when a start is detected on the receiver input. Its purpose is primarily to allow wake-up from Deep-sleep or Power-down mode immediately when a start is detected. Cleared by software.	0	W1

Table 162. USART Status register (STAT, address 0x4006 4008 (USART0), 0x4006 8008 (USART1), 0x4006 C008 (USART2)) bit description

Bit	Symbol	Description	Reset value	Access [1]
13	FRAMERRINT	Framing Error interrupt flag. This flag is set when a character is received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.	0	W1
14	PARITYERRINT	Parity Error interrupt flag. This flag is set when a parity error is detected in a received character..	0	W1
15	RXNOISEINT	Received Noise interrupt flag. Three samples of received data are taken in order to determine the value of each received data bit, except in synchronous mode. This acts as a noise filter if one sample disagrees. This flag is set when a received data bit contains one disagreeing sample. This could indicate line noise, a baud rate or character format mismatch, or loss of synchronization during data reception.	0	W1
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA	NA

[1] RO = Read-only, W1 = write 1 to clear.

15.6.4 USART Interrupt Enable read and set register

The INTENSET register is used to enable various USART interrupt sources. Enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The INTENCLR register is used to clear bits in this register.

Table 163. USART Interrupt Enable read and set register (INTENSET, address 0x4006 400C (USART0), 0x4006 800C (USART1), 0x4006 C00C (USART2)) bit description

Bit	Symbol	Description	Reset Value
0	RXRDYEN	When 1, enables an interrupt when there is a received character available to be read from the RXDATA register.	0
1	-	Reserved. Read value is undefined, only zero should be written.	NA
2	TXRDYEN	When 1, enables an interrupt when the TXDATA register is available to take another character to transmit.	0
4:3	-	Reserved. Read value is undefined, only zero should be written.	NA
5	DELTACTSEN	When 1, enables an interrupt when there is a change in the state of the CTS input.	0
6	TXDISINTEN	When 1, enables an interrupt when the transmitter is fully disabled as indicated by the TXDISINT flag in STAT. See description of the TXDISINT bit for details.	0
7	-	Reserved. Read value is undefined, only zero should be written.	NA
8	OVERRUNEN	When 1, enables an interrupt when an overrun error occurred.	0
10:9	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 163. USART Interrupt Enable read and set register (INTENSET, address 0x4006 400C (USART0), 0x4006 800C (USART1), 0x4006 C00C (USART2)) bit description

Bit	Symbol	Description	Reset Value
11	DELTARXBRKEN	When 1, enables an interrupt when a change of state has occurred in the detection of a received break condition (break condition asserted or deasserted).	0
12	STARTEN	When 1, enables an interrupt when a received start bit has been detected.	0
13	FRAMERREN	When 1, enables an interrupt when a framing error has been detected.	0
14	PARITYERREN	When 1, enables an interrupt when a parity error has been detected.	0
15	RXNOISEEN	When 1, enables an interrupt when noise is detected. See description of the RXNOISEINT bit in Table 162.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

15.6.5 USART Interrupt Enable Clear register

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

Table 164. USART Interrupt Enable clear register (INTENCLR, address 0x4006 4010 (USART0), 0x4006 8010 (USART1), 0x4006 C010 (USART2)) bit description

Bit	Symbol	Description	Reset Value
0	RXRDYCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
1	-	Reserved. Read value is undefined, only zero should be written.	NA
2	TXRDYCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
4:3	-	Reserved. Read value is undefined, only zero should be written.	NA
5	DELTACTSCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
6	TXDISINTCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
7	-	Reserved. Read value is undefined, only zero should be written.	NA
8	OVERRUNCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
10:9	-	Reserved. Read value is undefined, only zero should be written.	NA
11	DELTARXBRKCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
12	STARTCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0

Table 164. USART Interrupt Enable clear register (INTENCLR, address 0x4006 4010 (USART0), 0x4006 8010 (USART1), 0x4006 C010 (USART2)) bit description

Bit	Symbol	Description	Reset Value
13	FRAMERRCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
14	PARITYERRCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
15	RXNOISECLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

15.6.6 USART Receiver Data register

The RXDATA register contains the last character received before any overrun.

Remark: Reading this register changes the status flags in the RXDATASTAT register.

Table 165. USART Receiver Data register (RXDATA, address 0x4006 4014 (USART0), 0x4006 8014 (USART1), 0x4006 C014 (USART2)) bit description

Bit	Symbol	Description	Reset Value
8:0	RXDAT	The USART Receiver Data register contains the next received character. The number of bits that are relevant depends on the USART configuration settings.	0
31:9	-	Reserved, the value read from a reserved bit is not defined.	NA

15.6.7 USART Receiver Data with Status register

The RXDATASTAT register contains the next complete character to be read and its relevant status flags. This allows getting all information related to a received character with one 16-bit read.

Remark: Reading this register changes the status flags.

Table 166. USART Receiver Data with Status register (RXDATASTAT, address 0x4006 4018 (USART0), 0x4006 8018 (USART1), 0x4006 C018 (USART2)) bit description

Bit	Symbol	Description	Reset Value
8:0	RXDAT	The USART Receiver Data register contains the next received character. The number of bits that are relevant depends on the USART configuration settings.	0
12:9	-	Reserved, the value read from a reserved bit is not defined.	NA
13	FRAMERR	Framing Error status flag. This bit is valid when there is a character to be read in the RXDATA register and reflects the status of that character. This bit will set when the character in RXDAT was received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.	0

Table 166. USART Receiver Data with Status register (RXDATASTAT, address 0x4006 4018 (USART0), 0x4006 8018 (USART1), 0x4006 C018 (USART2)) bit description

Bit	Symbol	Description	Reset Value
14	PARITYERR	Parity Error status flag. This bit is valid when there is a character to be read in the RXDATA register and reflects the status of that character. This bit will be set when a parity error is detected in a received character.	0
15	RXNOISE	Received Noise flag. See description of the RxNoiseInt bit in Table 162.	0
31:16	-	Reserved, the value read from a reserved bit is not defined.	NA

15.6.8 USART Transmitter Data Register

The TXDATA register is written in order to send data via the USART transmitter. That data will be transferred to the transmit shift register when it is available, and another character may then be written to TXDATA.

Table 167. USART Transmitter Data Register (TXDATA, address 0x4006 401C (USART0), 0x4006 801C (USART1), 0x4006 C01C (USART2)) bit description

Bit	Symbol	Description	Reset Value
8:0	TXDAT	Writing to the USART Transmit Data Register causes the data to be transmitted as soon as the transmit shift register is available and any conditions for transmitting data are met: CTS low (if CTSEN bit = 1), TXDIS bit = 0.	0
31:9	-	Reserved. Only zero should be written.	NA

15.6.9 USART Baud Rate Generator register

The Baud Rate Generator is a simple 16-bit integer divider controlled by the BRG register. The BRG register contains the value used to divide the base clock in order to produce the clock used for USART internal operations.

A 16-bit value allows producing standard baud rates from 300 baud and lower at the highest frequency of the device, up to 921,600 baud from a base clock as low as 14.7456 MHz.

Typically, the baud rate clock is 16 times the actual baud rate. This overclocking allows for centering the data sampling time within a bit cell, and for noise reduction and detection by taking three samples of incoming data.

Details on how to select the right values for BRG can be found later in this chapter, see Section 15.7.1.

Remark: If software needs to change the baud rate, the following sequence should be used: 1) Make sure the USART is not currently sending or receiving data. 2) Disable the USART by writing a 0 to the Enable bit (0 may be written to the entire registers). 3) Write the new BRGVAL. 4) Write to the CFG register to set the Enable bit to 1.

Table 168. USART Baud Rate Generator register (BRG, address 0x4006 4020 (USART0), 0x4006 8020 (USART1), 0x4006 C020 (USART2)) bit description

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

15.6.10 USART Interrupt Status register

The read-only INTSTAT register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See Table 162 for detailed descriptions of the interrupt flags.

Table 169. USART Interrupt Status register (INTSTAT, address 0x4006 4024 (USART0), 0x4006 8024 (USART1), 0x4006 C024 (USART2)) bit description

Bit	Symbol	Description	Reset Value
0	RXRDY	Receiver Ready flag.	0
1	-	Reserved. Read value is undefined, only zero should be written.	NA
2	TXRDY	Transmitter Ready flag.	1

Table 169. USART Interrupt Status register (INTSTAT, address 0x4006 4024 (USART0), 0x4006 8024 (USART1), 0x4006 C024 (USART2)) bit description

Bit	Symbol	Description	Reset Value
4:3	-	Reserved. Read value is undefined, only zero should be written.	NA
5	DELTACTS	This bit is set when a change in the state of the CTS input is detected.	0
6	TXDISINT	Transmitter Disabled Interrupt flag.	0
7	-	Reserved. Read value is undefined, only zero should be written.	NA
8	OVERRUNINT	Overrun Error interrupt flag.	0
10:9	-	Reserved. Read value is undefined, only zero should be written.	NA
11	DELTARXBRK	This bit is set when a change in the state of receiver break detection occurs.	0
12	START	This bit is set when a start is detected on the receiver input.	0
13	FRAMERRINT	Framing Error interrupt flag.	0
14	PARITYERRINT	Parity Error interrupt flag.	0
15	RXNOISEINT	Received Noise interrupt flag.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

15.7 Functional description

15.7.1 Clocking and Baud rates

In order to use the USART, clocking details must be defined such as setting up the BRG, and typically also setting up the FRG. See Figure 27.

15.7.1.1 Fractional Rate Generator (FRG)

The Fractional Rate Generator can be used to obtain more precise baud rates when the peripheral clock is not a good multiple of standard (or otherwise desirable) baud rates.

The FRG is typically set up to produce an integer multiple of the highest required baud rate, or a very close approximation. The BRG is then used to obtain the actual baud rate needed.

The FRG register controls the USART Fractional Rate Generator, which provides the base clock for the USART. The Fractional Rate Generator creates a lower rate output clock by suppressing selected input clocks. When not needed, the value of 0 can be set for the FRG, which will then not divide the input clock.

The FRG output clock is defined as the inputs clock divided by 1 + (MULT / 256), where MULT is in the range of 1 to 255. This allows producing an output clock that ranges from the input clock divided by 1+1/256 to 1+255/256 (just more than 1 to just less than 2). Any further division can be done specific to each USART block by the integer BRG divider contained in each USART.

The base clock produced by the FRG cannot be perfectly symmetrical, so the FRG distributes the output clocks as evenly as is practical. Since the USART normally uses 16x overclocking, the jitter in the fractional rate clock in these cases tends to disappear in the ultimate USART output.

For setting up the fractional divider use the following registers:

Table 23 “USART fractional generator divider value register (UARTFRGDIV, address 0x4004 80F0) bit description”

Table 24 “USART fractional generator multiplier value register (UARTFRGMULT, address 0x4004 80F4) bit description”

For details see Section 15.3.1 “Configure the USART clock and baud rate”.

15.7.1.2 Baud Rate Generator (BRG)

The Baud Rate Generator (see Section 15.6.9) is used to divide the base clock to produce a rate 16 times the desired baud rate. Typically, standard baud rates can be generated by integer divides of higher baud rates.

15.7.1.3 Baud rate calculations

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

15.7.2 Synchronous mode

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

15.7.3 Flow control

The USART supports both hardware and software flow control.

15.7.3.1 Hardware flow control

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

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

Figure 29 shows an overview of RTS and CTS within the USART.

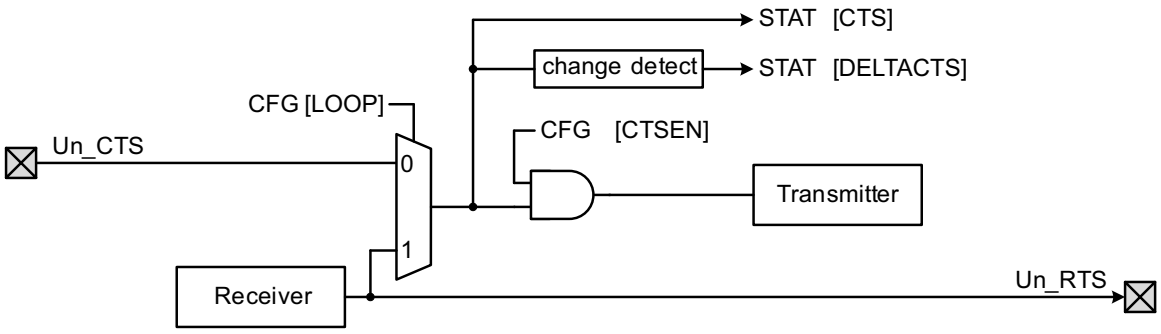


Fig 29. Hardware flow control using RTS and CTS

15.7.3.2 Software flow control

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

16.1 How to read this chapter

The I2C-bus interface is available on all parts.

Read this chapter if you want to understand the I2C operation and the software interface and want to learn how to use the I2C for wake-up from reduced power modes.

The LPC800 provides an on-chip ROM-based I2C API to configure and operate the I2C. See Table 258 “I2C API calls”.

16.2 Features

- Independent Master, Slave, and Monitor functions.
- Supports both Multi-master and Multi-master with Slave functions.
- Multiple I^2C slave addresses supported in hardware.
- One slave address can be selectively qualified with a bit mask or an address range in order to respond to multiple I^2C bus addresses.
- 10-bit addressing supported with software assist.
- Supports SMBus.

16.3 Basic configuration

Configure I2C using the following registers:

- In the SYSAHBCLKCTRL register, set bit 5 (Table 18) to enable the clock to the register interface.
- Clear the I2C peripheral reset using the PRESETCTRL register (Table 7).
- Enable/disable the I2C interrupt in interrupt slots #8 in the NVIC.
- Configure the I2C pin functions through the switch matrix. See Section 16.4.
- The peripheral clock for the I2C is the system clock (see Figure 30).

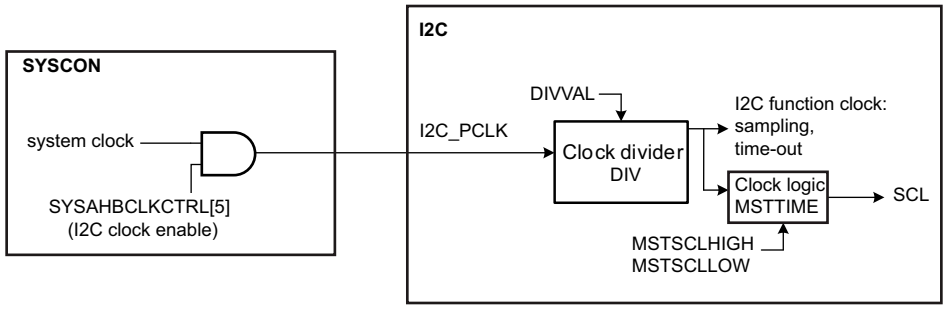


Fig 30. I2C clocking

16.3.1 I2C transmit/receive in master mode

In this example, the LPC800 I2C is configured as the master. The master sends 8 bits to the slave and then receives 8 bits from the slave. The system clock is set to 30 MHz and the bit rate is about 400 KHz. Therefore, you can select any pin for the I2C0_SCL and I2C0_SDA functions. Special open-drain I2C pads are optional. The transmission of the address and data bits is controlled by the state of the MSTPENDING status bit. Whenever the status is Master pending, the master can read or write to the MSTDAT register and go to the next step of the transmission protocol by writing to the MSTCTRL register.

Configure the pins:

- Select two pins for I2C0_SCL and I2C0_SDA through the switch matrix. See Table 170.
- In the IOCON register for the selected pins, disable the internal pull-up if using a standard digital I/O pin.

Configure the I2C bit rate:

- Divide the system clock (= I2C_PCLK) by a factor of 2. See Table 179 “I^2C Clock Divider register (DIV, address 0x4005 0014) bit description”.
- Set the SCL high and low times to 2 clock cycles each. This is the default. See Table 182 “Master Time register (MSTTIME, address 0x4005 0024) bit description”. The result is an SCL clock of 375 kHz.

Configure the LPC800 I2C as master: Set the MSTEN bit to 1 in the CFG register. See Table 172.

Write data to the slave:

1. Write the slave address with the $\overline{RW}$ bit set to 0 to the Master data register MSTDAT. See Table 183.
2. Start the transmission by setting the MSTSTART bit to 1 in the Master control register. See Table 181. The following happens:
   - The pending status is cleared and the I2C bus is busy.
   - The I2C master sends the start bit and address with the $\overline{RW}$ bit to the slave.
3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
4. Write 8 bits of data to the MSTDAT register.
5. Continue with the transmission of the data by setting the MSTCONT bit to 1 in the Master control register. See Table 181. The following happens:
   - The pending status is cleared and the I2C bus is busy.
   - The I2C master sends the data bits to the slave address.
6. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
7. Stop the transmission by setting the MSTSTOP bit to 1 in the Master control register. See Table 181.

Read data from the slave:

1. Write the slave address with the $\overline{RW}$ bit set to 1 to the Master data register MSTDAT. See Table 183.

2. Start the transmission by setting the MSTSTART bit to 1 in the Master control register. See Table 181. The following happens:
   - The pending status is cleared and the I2C bus is busy.
   - The I2C master sends the start bit and address with the $\overline{RW}$ bit to the slave.
   - The slave sends 8 bit of data.
3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
4. Read 8 bits of data from the MSTDAT register.
5. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
6. Stop the transmission by setting the MSTSTOP bit to 1 in the Master control register. See Table 181.

Write data to the slave and read back 2 bytes of data from the slave:

1. Write the slave address with the $\overline{RW}$ bit set to 0 to the Master data register MSTDAT. See Table 183.
2. Start the transmission by setting the MSTSTART bit to 1 in the Master control register. See Table 181. The following happens:
   - The pending status is cleared and the I2C bus is busy.
   - The I2C master sends the start bit and address with the $\overline{RW}$ bit to the slave.
3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
4. Write 8 bits of data to the MSTDAT register.
5. Continue with the transmission of the data by setting the MSTCONTINUE bit to 1 in the Master control register. See Table 181. The following happens:
   - The pending status is cleared and the I2C bus is busy.
   - The I2C master sends the data bits to the slave address.
6. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
7. Write the slave address with the $\overline{RW}$ bit set to 1 to the Master data register MSTDAT. See Table 183.
8. Re-start the transmission setting the MSTSTART bit to 1 in the Master control register. See Table 181. The following happens:
   - The pending status is cleared and the I2C bus is busy.
   - The I2C master sends the start bit and address with the $\overline{RW}$ bit to the slave.
   - The slave sends 8 bit of data.
9. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
10. Read the first byte of data from the MSTDAT register.
11. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
12. Repeat reading data from the slave by setting the MSTCONTINUE bit to 1 in the Master control register.
13. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
14. Read the second byte of data from the MSTDAT register.
15. Stop the transmission by setting the MSTSTOP bit to 1 in the Master control register. See Table 181.

16.3.2 Configure the I2C for wake-up

In sleep mode, any activity on the I2C-bus that triggers an I2C interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the I2C clock I2C_PCLK remains active in sleep mode, the I2C can wake up the part independently of whether the I2C block is configured in master or slave mode.

In Deep-sleep or Power-down mode, the I2C clock is turned off as are all peripheral clocks. However, if the I2C is configured in slave mode and an external master on the I2C-bus provides the clock signal, the I2C block can create an interrupt asynchronously. This interrupt, if enabled in the NVIC, the STARTERP1 register, and in the I2C block's INTENCLR register, can then wake up the core.

16.3.2.1 Wake-up from Sleep mode

- Enable the I2C interrupt in the NVIC.
- Enable the I2C wake-up event in the I2C INTENSET register. Wake-up on any enabled interrupts is supported (see the INTENSET register). Examples are the following events:
  - Master pending
  - Change to idle state
  - Start/stop error
  - Slave pending
  - Address match (in slave mode)
  - Data available/ready

16.3.2.2 Wake-up from Deep-sleep and Power-down modes

- Enable the I2C interrupt in the NVIC.
- Enable the I2C interrupt in the STARTERP1 register in the SYSCON block to create the interrupt signal asynchronously while the core and the peripheral are not clocked. See Table 34 "Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description".
- In the PDAWAKE register, configure all peripherals that need to be running when the part wakes up.
- Configure the I2C in slave mode
- Enable the I2C the interrupt in the I2C INTENCLR register which configures the interrupt as wake-up event. Examples are the following events:
  - Slave deselect
  - Slave pending (wait for read, write, or ACK)
  - Address match
  - Data available/ready for the monitor

16.4 Pin description

The I2C pins are movable pin functions and are assigned to pins on the LPC800 packages through the switch matrix. You have two choices to connect the I2C pins:

1. Connect to special I2C open-drain pins (PIO0_10 and PIO0_11).
2. Connect to any other pin that can host a movable function.

When the I^2C function is connected to specialized I^2C pins, it supports the full I^2C-bus specification up to Fast Mode Plus (up to 1 MHz I^2C).

When the I^2C function is connected to standard pins that are set to open-drain mode, a functional I^2C-bus can be used in this way, but some aspects of the I^2C-bus specification may not be met. This can have an impact on the bus speed, noise filtering, and the capability of powering down the device without affecting the bus.

See Section 9.3.1 “Connect an internal signal to a package pin” to assign the I2C pins to any pin on the LPC800 package.

Table 170. I2C-bus pin description

Function	Type	Pin	Description	SWM register	Reference
I2C0_SCL	I/O	any; use pin PIO0_10 or PIO0_11 for compatibility with the full I2C-bus specification.	I2C0 serial clock.	PINASSIGN8	Table 105
I2C0_SDA	I/O	any; use pin PIO0_10 or PIO0_11 for compatibility with the full I2C-bus specification.	I2C0 serial data.	PINASSIGN7	Table 104

16.5 General description

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

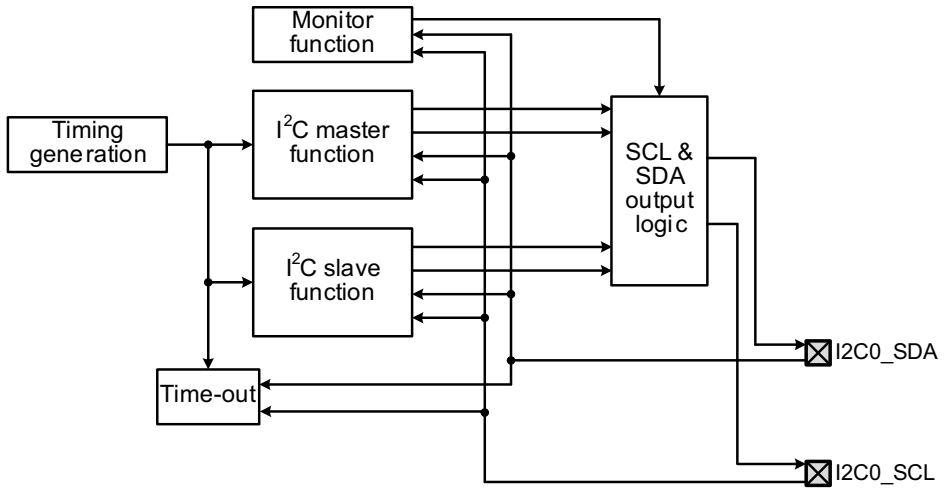


Fig 31. I2C block diagram

16.6 Register description

The register functions can be grouped as follows:

- Common registers:
  - Table 172 “I2C Configuration register (CFG, address 0x4005 0000) bit description”

- Table 173 “I2C Status register (STAT, address 0x4005 0004) bit description”
- Table 180 “I2C Interrupt Status register (INTSTAT, address 0x4005 0018) bit description”
- Table 176 “Interrupt Enable Set and read register (INTENSET, address 0x4005 0008) bit description”
- Table 177 “Interrupt Enable Clear register (INTENCLR, address 0x4005 000C) bit description”
- Table 178 “time-out register (TIMEOUT, address 0x4005 0010) bit description”
- Table 179 “I2C Clock Divider register (DIV, address 0x4005 0014) bit description”

- Master function registers:
  - Table 181 “Master Control register (MSTCTL, address 0x4005 0020) bit description”
  - Table 182 “Master Time register (MSTTIME, address 0x4005 0024) bit description”
  - Table 183 “Master Data register (MSTDAT, address 0x4005 0028) bit description”
- Slave function registers:
  - Table 184 “Slave Control register (SLVCTL, address 0x4005 0040) bit description”
  - Table 184 “Slave Control register (SLVCTL, address 0x4005 0040) bit description”
  - Table 186 “Slave Address registers (SLVADR[0:3], address 0x4005 0048 (SLVADR0) to 0x4005 0054 (SLVADR3)) bit description”
  - Table 187 “Slave address Qualifier 0 register (SLVQUAL0, address 0x4005 0058) bit description”
- Monitor function register: Table 188 “Monitor data register (MONRXDAT, address 0x4005 0080) bit description”

Table 171: Register overview: I2C (base address 0x4005 0000)

Name	Access	Offset	Description	Reset value	Reference
CFG	R/W	0x00	Configuration for shared functions.	0	Table 172
STAT	R/W	0x04	Status register for Master, Slave, and Monitor functions.	0x00080 1	Table 173
INTENSET	R/W	0x08	Interrupt Enable Set and read register.	0	Table 176
INTENCLR	W	0x0C	Interrupt Enable Clear register.	NA	Table 177
TIMEOUT	R/W	0x10	Time-out value register.	0xFFFF	Table 178
DIV	R/W	0x14	Clock pre-divider for the entire I2C block. This determines what time increments are used for the MSTTIME and SLVTIME registers.	0	Table 179
INTSTAT	R	0x18	Interrupt Status register for Master, Slave, and Monitor functions.	0	Table 180
MSTCTL	R/W	0x20	Master control register.	0	Table 181
MSTTIME	R/W	0x24	Master timing configuration.	0x77	Table 182
MSTDAT	R/W	0x28	Combined Master receiver and transmitter data register.	NA	Table 183
SLVCTL	R/W	0x40	Slave control register.	0	Table 184
SLVDAT	R/W	0x44	Combined Slave receiver and transmitter data register.	NA	Table 185
SLVADR0	R/W	0x48	Slave address 0.	0x01	Table 186
SLVADR1	R/W	0x4C	Slave address 1.	0x01	Table 186
SLVADR2	R/W	0x50	Slave address 2.	0x01	Table 186
SLVADR3	R/W	0x54	Slave address 3.	0x01	Table 186
SLVQUAL0	R/W	0x58	Slave Qualification for address 0.	0	Table 187
MONRXDAT	RO	0x80	Monitor receiver data register.	0	Table 188

16.6.1 I2C Configuration register

The CFG register contains mode settings that apply to Master, Slave, and Monitor functions.

Table 172. I2C Configuration register (CFG, address 0x4005 0000) bit description

Bit	Symbol	Value	Description	Reset Value
0	MSTEN		Master Enable. When disabled, configurations settings for the Master function are not changed, but the Master function is internally reset.	0
		0	Disabled. The I2C Master function is disabled.	
		1	Enabled. The I2C Master function is enabled.	
1	SLVEN		Slave Enable. When disabled, configurations settings for the Slave function are not changed, but the Slave function is internally reset.	0
		0	Disabled. The I2C slave function is disabled.	
		1	Enabled. The I2C slave function is enabled.	

Table 172. I2C Configuration register (CFG, address 0x4005 0000) bit description

Bit	Symbol	Value	Description	Reset Value
2	MONEN		Monitor Enable. When disabled, configurations settings for the Monitor function are not changed, but the Monitor function is internally reset.	0
		0	Disabled. The I^2C monitor function is disabled.	
		1	Enabled. The I^2C monitor function is enabled.	
3	TIMEOUTEN		I^2C bus Time-out Enable. When disabled, the time-out function is internally reset.	0
		0	Disabled. Time-out function is disabled.	
		1	Enabled. Time-out function is enabled. Both types of time-out flags will be generated and will cause interrupts if they are enabled. Typically, only one time-out will be used in a system.	
4	MONCLKSTR		Monitor function Clock Stretching.	0
		0	Disabled. The monitor function will not perform clock stretching. Software may not always be able to read data provided by the monitor function before it is overwritten. This mode may be used when non-invasive monitoring is critical.	
		1	Enabled. The monitor function will perform clock stretching in order to ensure that software can read all incoming data supplied by the monitor function.	
31:5	-		Reserved. Read value is undefined, only zero should be written.	NA

16.6.2 I2C Status register

The STAT register provides status flags and state information about all of the functions of the I2C block. Some information in this register is read-only and some flags can be cleared by writing a 1 to them.

Access to bits in this register varies. RO = Read-only, W1 = write 1 to clear.

Details on the master and slave states described in the MSTSTATE and SLVSTATE bits in this register are listed in Table 174 and Table 175.

Table 173. I2C Status register (STAT, address 0x4005 0004) bit description

Bit	Symbol	Value	Description	Reset value	Access
0	MSTPENDING		Master Pending. Indicates that the Master is waiting to continue communication on the I2C bus (pending) or is idle. When the master is pending, the MSTSTATE bits indicate what type of software service if any the master expects. This flag will cause an interrupt when set if, enabled via the INTENSET register. If the master is in the idle state, and no communication is needed, mask this interrupt.	1	RO
		0	In progress. Communication is in progress and the Master function is busy and cannot currently accept a command.		
		1	Pending. The Master function needs software service or is in the idle state. If the master is not in the idle state, it is waiting to receive or transmit data or the NACK bit.		
3:1	MSTSTATE		Master State code. The master state code reflects the master state when the MSTPENDING bit is set, that is the master is pending or in the idle state. Each value of this field indicates a specific required service for the Master function. All other values are reserved.	0	RO
		0x0	Idle. The Master function is available to be used for a new transaction.		
		0x1	Receive ready. Received data available (Master Receiver mode). Address plus Read was previously sent and Acknowledged by slave.		
		0x2	Transmit ready. Data can be transmitted (Master Transmitter mode). Address plus Write was previously sent and Acknowledged by slave.		
		0x3	NACK Address. Slave NACKed address.		
		0x4	NACK Data. Slave NACKed transmitted data.		
4	MSTARBLOSS		Master Arbitration Loss flag. This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically a 1 is written to MSTCONTINUE.	0	W1
		0	No loss. No Arbitration Loss has occurred.		
		1	Arbitration loss. The Master function has experienced an Arbitration Loss. At this point, the Master function has already stopped driving the bus and gone to an idle state. Software can respond by doing nothing, or by sending a Start in order to attempt to gain control of the bus when it next becomes idle.		
5	-		Reserved. Read value is undefined, only zero should be written.	NA	NA

Table 173. I²C Status register (STAT, address 0x4005 0004) bit description *…continued*

Bit	Symbol	Value	Description	Reset value	Access
6	MSTSTSTPERR		Master Start/Stop Error flag. This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically a 1 is written to MstContinue.	0	W1
		0	No Start/Stop Error has occurred.		
		1	Start/stop error has occurred. The Master function has experienced a Start/Stop Error. A Start or Stop was detected at a time when it is not allowed by the I²C specification. The Master interface has stopped driving the bus and gone to an idle state, no action is required. A request for a Start could be made, or software could attempt to insure that the bus has not stalled.		
7	-		Reserved. Read value is undefined, only zero should be written.	NA	NA
8	SLVPENDING		Slave Pending. Indicates that the Slave function is waiting to continue communication on the I2C-bus and needs software service. This flag will cause an interrupt when set if enabled via INTENSET. The SLVPENDING flag is read-only and is automatically cleared when a 1 is written to the SLVCONTINUE bit in the SLVCTL register.	0	RO
		0	In progress. The Slave function does not currently need service.		
		1	Pending. The Slave function needs service. Information on what is needed can be found in the adjacent SLVSTATE field.		
10:9	SLVSTATE		Slave State code. Each value of this field indicates a specific required service for the Slave function. All other values are reserved.	0	RO
		0x0	Slave address.. Address plus R/W received. At least one of the four slave addresses has been matched by hardware.		
		0x1	Slave receive. Received data is available (Slave Receiver mode).		
		0x2	Slave transmit. Data can be transmitted (Slave Transmitter mode).		
		0x3	Reserved.		
11	SLVNOTSTR		Slave Not Stretching. Indicates when the slave function is stretching the I²C clock. This is needed in order to gracefully invoke Deep Sleep or Power-down modes during slave operation. This read-only flag reflects the slave function status in real time.	1	RO
		0	Stretching. The slave function is currently stretching the I²C bus clock. Deep-Sleep or Power-down mode cannot be entered at this time.		
		1	Not stretching. The slave function is not currently stretching the I²C bus clock. Deep-sleep or Power-down mode could be entered at this time.		
13:12	SLVIDX		Slave address match Index. This field is valid when the I²C slave function has been selected by receiving an address that matches one of the slave addresses defined by any enabled slave address registers, and provides an identification of the address that was matched. It is possible that more than one address could be matched, but only one match can be reported here.	0	RO
		0x0	Slave address 0 was matched.		
		0x1	Slave address 1 was matched.		
		0x2	Slave address 2 was matched.		
		0x3	Slave address 3 was matched.		

Table 173. I²C Status register (STAT, address 0x4005 0004) bit description *…continued*

Bit	Symbol	Value	Description	Reset value	Access
14	SLVSEL		Slave selected flag. SLVSEL is set after an address match when software tells the Slave function to acknowledge the address. It is cleared when another address cycle presents an address that does not match an enabled address on the Slave function, when slave software decides to NACK a matched address, or when there is a Stop detected on the bus. SLVSEL is not cleared if software NACKs data.	0	RO
		0	Not selected. The Slave function is not currently selected.		
		1	Selected. The Slave function is currently selected.		
15	SLVDESEL		Slave Deselected flag. This flag will cause an interrupt when set if enabled via INTENSET. This flag can be cleared by writing a 1 to this bit.	0	W1
		0	Not deselected. The Slave function has not become deselected. This does not mean that it is currently selected. That information can be found in the SLVSEL flag.		
		1	Deselected. The Slave function has become deselected. This is specifically caused by the SLVSEL flag changing from 1 to 0. See the description of SLVSEL for details on when that event occurs.		
16	MONRDY		Monitor Ready. This flag is cleared when the MONRXDAT register is read.	0	RO
		0	No data. The Monitor function does not currently have data available.		
		1	Data waiting. The Monitor function has data waiting to be read.		
17	MONOV		Monitor Overflow flag.	0	W1
		0	No overrun. Monitor data has not overrun.		
		1	Overrun. A Monitor data overrun has occurred. This can only happen when Monitor clock stretching not enabled via the MONCLKSTR bit in the CFG register. Writing 1 to this bit clears the flag.		
18	MONACTIVE		Monitor Active flag. This flag indicates when the Monitor function considers the I²C bus to be active. Active is defined here as when some Master is on the bus: a bus Start has occurred more recently than a bus Stop.	0	RO
		0	Inactive. The Monitor function considers the I²C bus to be inactive.		
		1	Active. The Monitor function considers the I²C bus to be active.		
19	MONIDLE		Monitor Idle flag. This flag is set when the Monitor function sees the I²C bus change from active to inactive. This can be used by software to decide when to process data accumulated by the Monitor function. This flag will cause an interrupt when set if enabled via the INTENSET register . The flag can be cleared by writing a 1 to this bit.	0	W1
		0	Not idle. The I²C bus is not idle, or this flag has been cleared by software.		
		1	Idle. The I²C bus has gone idle at least once since the last time this flag was cleared by software.		
23:20	-		Reserved. Read value is undefined, only zero should be written.	NA	NA

Table 173. I²C Status register (STAT, address 0x4005 0004) bit description *…continued*

Bit	Symbol	Value	Description	Reset value	Access
24	EVENTTIMEOUT		Event Time-out Interrupt flag. Indicates when the time between events has been longer than the time specified by the TIMEOUT register. Events include Start, Stop, and clock edges. The flag is cleared by writing a 1 to this bit. No time-out is created when the I2C bus is idle.	0	W1
		0	No time-out. I²C bus events have not caused a time-out.		
		1	Event time-out. The time between I²C bus events has been longer than the time specified by the I2C TIMEOUT register.		
25	SCLTIMEOUT		SCL Time-out Interrupt flag. Indicates when SCL has remained low longer than the time specific by the TIMEOUT register. The flag is cleared by writing a 1 to this bit.	0	W1
		0	No time-out. SCL low time has not caused a time-out.		
		1	Time-out. SCL low time has caused a time-out.		
31:26	-		Reserved. Read value is undefined, only zero should be written.	NA	NA

Table 174: Master function state codes (MSTSTATE)

MstState	Description	Actions
0	**Idle.** The Master function is available to be used for a new transaction.	Send a Start or disable MSTPENDING interrupt if the Master function is not needed currently.
1	**Received data is available (Master Receiver mode).** Address plus Read was previously sent and Acknowledged by slave.	Read data and either continue, send a Stop, or send a Repeated Start.
2	**Data can be transmitted (Master Transmitter mode).** Address plus Write was previously sent and Acknowledged by slave.	Send data and continue, or send a Stop or Repeated Start.
3	**Slave NACKed address.**	Send a Stop or Repeated Start.
4	**Slave NACKed transmitted data.**	Send a Stop or Repeated Start.

Table 175: Slave function state codes (SLVSTATE)

SlvState	Description	Actions
0	**Address plus R/W received.** At least one of the 4 slave addresses has been matched by hardware.	Software can further check the address if needed, for instance if a subset of addresses qualified by SLVQUAL0 is to be used. Software can ACK or NACK the address by writing 1 to either SLVCONTINUE or SLVNACK. Also see Section 16.7.3 regarding 10-bit addressing.
1	**Received data is available (Slave Receiver mode).**	Read data reply with an ACK or a NACK.
2	**Data can be transmitted (Slave Transmitter mode).**	Send data.
3	Reserved.	-

16.6.3 Interrupt Enable Set and read register

The INTENSET register controls which I^2C status flags generate interrupts. Writing a 1 to a bit position in this register enables an interrupt in the corresponding position in the STAT register, if an interrupt is supported there. Reading INTENSET indicates which interrupts are currently enabled.

Table 176. Interrupt Enable Set and read register (INTENSET, address 0x4005 0008) bit description

Bit	Symbol	Value	Description	Reset value
0	MSTPENDINGEN		Master Pending interrupt Enable.	0
		0	The MstPending interrupt is disabled.	
		1	The MstPending interrupt is enabled.	
3:1	-		Reserved. Read value is undefined, only zero should be written.	NA
4	MSTARBLOSSEN		Master Arbitration Loss interrupt Enable.	0
		0	The MstArbLoss interrupt is disabled.	
		1	The MstArbLoss interrupt is enabled.	
5	-		Reserved. Read value is undefined, only zero should be written.	NA
6	MSTSTSTPERREN		Master Start/Stop Error interrupt Enable.	0
		0	The MstStStpErr interrupt is disabled.	
		1	The MstStStpErr interrupt is enabled.	
7	-		Reserved. Read value is undefined, only zero should be written.	NA
8	SLVPENDINGEN		Slave Pending interrupt Enable.	0
		0	The SlvPending interrupt is disabled.	
		1	The SlvPending interrupt is enabled.	
10:9	-		Reserved. Read value is undefined, only zero should be written.	NA
11	SLVNOTSTREN		Slave Not Stretching interrupt Enable.	0
		0	The SlvNotStr interrupt is disabled.	
		1	The SlvNotStr interrupt is enabled.	
14:12	-		Reserved. Read value is undefined, only zero should be written.	NA
15	SLVDESELEN		Slave Deselect interrupt Enable.	0
		0	The SlvDeSel interrupt is disabled.	
		1	The SlvDeSel interrupt is enabled.	
16	MONRDYEN		Monitor data Ready interrupt Enable.	0
		0	The MonRdy interrupt is disabled.	
		1	The MonRdy interrupt is enabled.	
17	MONOVEN		Monitor Overrun interrupt Enable.	0
		0	The MonOv interrupt is disabled.	
		1	The MonOv interrupt is enabled.	
18	-		Reserved. Read value is undefined, only zero should be written.	NA

Table 176. Interrupt Enable Set and read register (INTENSET, address 0x4005 0008) bit description

Bit	Symbol	Value	Description	Reset value
19	MONIDLEEN		Monitor Idle interrupt Enable.	0
		0	The MonIdle interrupt is disabled.	
		1	The MonIdle interrupt is enabled.	
23:20	-		Reserved. Read value is undefined, only zero should be written.	NA
24	EVENTTIMEOUTEN		Event time-out interrupt Enable.	0
		0	The Event time-out interrupt is disabled.	
		1	The Event time-out interrupt is enabled.	
25	SCLTIMEOUTEN		SCL time-out interrupt Enable.	0
		0	The SCL time-out interrupt is disabled.	
		1	The SCL time-out interrupt is enabled.	
31:26	-		Reserved. Read value is undefined, only zero should be written.	NA

16.6.4 Interrupt Enable Clear register

Writing a 1 to a bit position in INTENCLR clears the corresponding position in the INTENSET register, disabling that interrupt. INTENCLR is a write-only register.

Bits that do not correspond to defined bits in INTENSET are reserved and only zeroes should be written to them.

Table 177. Interrupt Enable Clear register (INTENCLR, address 0x4005 000C) bit description

Bit	Symbol	Description	Reset value
0	MSTPENDINGCLR	Master Pending interrupt clear. Writing 1 to this bit clears the corresponding bit in the INTENSET register if implemented.	0
3:1	-	Reserved. Read value is undefined, only zero should be written.	NA
4	MSTARBLOSSCLR	Master Arbitration Loss interrupt clear.	0
5	-	Reserved. Read value is undefined, only zero should be written.	NA
6	MSTSTSTPERRCLR	Master Start/Stop Error interrupt clear.	0
7	-	Reserved. Read value is undefined, only zero should be written.	NA
8	SLVPENDINGCLR	Slave Pending interrupt clear.	0
10:9	-	Reserved. Read value is undefined, only zero should be written.	NA
11	SLVNOTSTRCLR	Slave Not Stretching interrupt clear.	0
14:12	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 177. Interrupt Enable Clear register (INTENCLR, address 0x4005 000C) bit description *…continued*

Bit	Symbol	Description	Reset value
15	SLVDESELCLR	Slave Deselect interrupt clear.	0
16	MONRDYCLR	Monitor data Ready interrupt clear.	0
17	MONOVCLR	Monitor Overrun interrupt clear.	0
18	-	Reserved. Read value is undefined, only zero should be written.	NA
19	MONIDLECLR	Monitor Idle interrupt clear.	0
23:20	-	Reserved. Read value is undefined, only zero should be written.	NA
24	EVENTTIMEOUTCLR	Event time-out interrupt clear.	0
25	SCLTIMEOUTCLR	SCL time-out interrupt clear.	0
31:26	-	Reserved. Read value is undefined, only zero should be written.	NA

16.6.5 Time-out value register

The TIMEOUT register allows setting an upper limit to certain I^2C bus times, informing by status flag and/or interrupt when those times are exceeded.

Two time-outs are generated, and software can elect to use either of them.

1. EVENTTIMEOUT checks the time between bus events while the bus is not idle: Start, SCL rising, SCL falling, and Stop. The EVENTTIMEOUT status flag in the STAT register is set if the time between any two events becomes longer than the time configured in the TIMEOUT register. The EVENTTIMEOUT status flag can cause an interrupt if enabled to do so by the EVENTTIMEOUTEN bit in the INTENSET register.
2. SCLTIMEOUT checks only the time that the SCL signal remains low while the bus is not idle. The SCLTIMEOUT status flag in the STAT register is set if SCL remains low longer than the time configured in the TIMEOUT register. The SCLTIMEOUT status flag can cause an interrupt if enabled to do so by the SCLTIMEOUTEN bit in the INTENSET register. The SCLTIMEOUT can be used with the SMBus.

Also see Section 16.7.2 “Time-out”.

Table 178. time-out register (TIMEOUT, address 0x4005 0010) bit description

Bit	Symbol	Description	Reset value
3:0	TOMIN	Time-out time value, bottom four bits. These are hard-wired to 0xF. This gives a minimum time-out of 16 I^2C function clocks and also a time-out resolution of 16 I^2C function clocks.	0xF
15:4	TO	Time-out time value. Specifies the time-out interval value in increments of 16 I^2C function clocks, as defined by the CLKDIV register. To change this value while I^2C is in operation, disable all time-outs, write a new value to TIMEOUT, then re-enable time-outs. 0x000 = A time-out will occur after 16 counts of the I^2C function clock. 0x001 = A time-out will occur after 32 counts of the I^2C function clock. ... 0xFFF = A time-out will occur after 65,536 counts of the I^2C function clock.	0xFFF
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

16.6.6 I2C Clock Divider register

The CLKDIV register divides down the Peripheral Clock (PCLK) to produce the I^2C function clock that is used to time various aspects of the I^2C interface. The I^2C function clock is used for some internal operations in the I^2C block and to generate the timing required by the I^2C bus specification, some of which are user configured in the MSTTIME register for Master operation and the SLVTIME register for Slave operation.

See Section 16.7.1.1 “Rate calculations” for details on bus rate setup.

Table 179. I^2C Clock Divider register (DIV, address 0x4005 0014) bit description

Bit	Symbol	Description	Reset value
15:0	DIVVAL	This field controls how the clock (PCLK) is used by the I^2C functions that need an internal clock in order to operate. 0x0000 = PCLK is used directly by the I^2C function. 0x0001 = PCLK is divided by 2 before use by the I^2C function. 0x0002 = PCLK is divided by 3 before use by the I^2C function. ... 0xFFFF = PCLK is divided by 65,536 before use by the I^2C function.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

16.6.7 I2C Interrupt Status register

The INTSTAT register provides register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See Table 173 for detailed descriptions of the interrupt flags.

Table 180. I²C Interrupt Status register (INTSTAT, address 0x4005 0018) bit description

Bit	Symbol	Description	Reset value
0	MSTPENDING	Master Pending.	1
3:1	-	Reserved.	
4	MSTARBLOSS	Master Arbitration Loss flag.	0
5	-	Reserved. Read value is undefined, only zero should be written.	NA
6	MSTSTSTPERR	Master Start/Stop Error flag.	0
7	-	Reserved. Read value is undefined, only zero should be written.	NA
8	SLVPENDING	Slave Pending.	0
10:9	-	Reserved. Read value is undefined, only zero should be written.	NA
11	SLVNOTSTR	Slave Not Stretching status.	1
14:12	-	Reserved. Read value is undefined, only zero should be written.	NA
15	SLVDESEL	Slave Deselected flag.	0
16	MONRDY	Monitor Ready.	0
17	MONOV	Monitor Overflow flag.	0
18	-	Reserved. Read value is undefined, only zero should be written.	NA
19	MONIDLE	Monitor Idle flag.	0
23:20	-	Reserved. Read value is undefined, only zero should be written.	NA
24	EVENTTIMEOUT	Event time-out Interrupt flag.	0
25	SCLTIMEOUT	SCL time-out Interrupt flag.	0
31:26	-	Reserved. Read value is undefined, only zero should be written.	NA

16.6.8 Master Control register

The MSTCTL register contains bits that control various functions of the I²C Master interface. Only write to this register when the master is pending (MSTPENDING = 1 in the STAT register, Table 173).

Table 181. Master Control register (MSTCTL, address 0x4005 0020) bit description

Bit	Symbol	Value	Description	Reset value
0	MSTCONTINUE		Master Continue. This bit is write-only.	0
		0	No effect.	
		1	Continue. Informs the Master function to continue to the next operation. This must be done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation.	

Table 181. Master Control register (MSTCTL, address 0x4005 0020) bit description

Bit	Symbol	Value	Description	Reset value
1	MSTSTART		Master Start control. This bit is write-only.	0
		0	No effect.	
		1	Start. A Start will be generated on the I^2C bus at the next allowed time.	
2	MSTSTOP		Master Stop control. This bit is write-only.	0
		0	No effect.	
		1	Stop. A Stop will be generated on the I^2C bus at the next allowed time, preceded by a NACK to the slave if the master is receiving data from the slave (Master Receiver mode).	
31:2	-		Reserved. Read value is undefined, only zero should be written.	NA

16.6.9 Master Time

The MSTTIME register allows programming of certain times that may be controlled by the Master function. These include the clock (SCL) high and low times, repeated Start setup time, and transmitted data setup time.

The I2C clock pre-divider is described in Table 179.

Table 182. Master Time register (MSTTIME, address 0x4005 0024) bit description

Bit	Symbol	Value	Description	Reset value
2:0	MSTSCLLOW		Master SCL Low time. Specifies the minimum low time that will be asserted by this master on SCL. Other devices on the bus (masters or slaves) could lengthen this time. This corresponds to the parameter t_{LOW} in the I^2C bus specification. I^2C bus specification parameters t_{BUF} and $t_{SU;STA}$ have the same values and are also controlled by MSTSCLLOW.	0
		0x0	2 clocks. Minimum SCL low time is 2 clocks of the I^2C clock pre-divider.	
		0x1	3 clocks. Minimum SCL low time is 3 clocks of the I^2C clock pre-divider.	
		0x2	4 clocks. Minimum SCL low time is 4 clocks of the I^2C clock pre-divider.	
		0x3	5 clocks. Minimum SCL low time is 5 clocks of the I^2C clock pre-divider.	
		0x4	6 clocks. Minimum SCL low time is 6 clocks of the I^2C clock pre-divider.	
		0x5	7 clocks. Minimum SCL low time is 7 clocks of the I^2C clock pre-divider.	
		0x6	8 clocks. Minimum SCL low time is 8 clocks of the I^2C clock pre-divider.	
		0x7	9 clocks. Minimum SCL low time is 9 clocks of the I^2C clock pre-divider.	

Table 182. Master Time register (MSTTIME, address 0x4005 0024) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
6:4	MSTSCLHIGH		Master SCL High time. Specifies the minimum high time that will be asserted by this master on SCL. Other masters in a multi-master system could shorten this time. This corresponds to the parameter t_{HIGH} in the I^2C bus specification. I^2C bus specification parameters $t_{SU;STO}$ and $t_{HD;STA}$ have the same values and are also controlled by MSTSCLHIGH.	0
		0x0	2 clocks. Minimum SCL high time is 2 clock of the I^2C clock pre-divider.	
		0x1	3 clocks. Minimum SCL high time is 3 clocks of the I^2C clock pre-divider .	
		0x2	4 clocks. Minimum SCL high time is 4 clock of the I^2C clock pre-divider.	
		0x3	5 clocks. Minimum SCL high time is 5 clock of the I^2C clock pre-divider.	
		0x4	6 clocks. Minimum SCL high time is 6 clock of the I^2C clock pre-divider.	
		0x5	7 clocks. Minimum SCL high time is 7 clock of the I^2C clock pre-divider.	
		0x6	8 clocks. Minimum SCL high time is 8 clock of the I^2C clock pre-divider.	
		0x7	9 clocks. Minimum SCL high time is 9 clocks of the I^2C clock pre-divider.	
31:7	-		Reserved. Read value is undefined, only zero should be written.	NA

16.6.10 Master Data register

The MSTDAT register provides the means to read the most recently received data for the Master function, and to transmit data using the Master function.

Table 183. Master Data register (MSTDAT, address 0x4005 0028) bit description

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

16.6.11 Slave Control register

The SLVCTL register contains bits that control various functions of the I^2C Slave interface. Only write to this register when the slave is pending (SLVPENDING =1 in the STAT register, Table 173).

Table 184. Slave Control register (SLVCTL, address 0x4005 0040) bit description

Bit	Symbol	Value	Description	Reset Value
0	SLVCONTINUE		Slave Continue.	0
		0	No effect.	
		1	Continue. Informs the Slave function to continue to the next operation. This must done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation.	
1	SLVNACK		Slave NACK.	0
		0	No effect.	
		1	NACK. Causes the Slave function to NACK the master when the slave is receiving data from the master (Slave Receiver mode).	
31:2	-		Reserved. Read value is undefined, only zero should be written.	NA

16.6.12 Slave Data register

The SLVDAT register provides the means to read the most recently received data for the Slave function and to transmit data using the Slave function.

Table 185. Slave Data register (SLVDAT, address 0x4005 0044) bit description

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

16.6.13 Slave Address registers

The SLVADR[0:3] registers allow enabling and defining one of the addresses that can be automatically recognized by the I^2C slave hardware. The value in the SLVADR0 register is qualified by the setting of the SLVQUAL0 register.

When the slave address is compared to the receive address, the compare can be affected by the setting of the SLVQUAL0 register (see Section 16.6.14).

The I^2C slave function has 4 address comparators. The additional 3 address comparators do not include the address qualifier feature. For handling of the general call address, one of the 4 address registers can be programmed to respond to address 0.

Table 186. Slave Address registers (SLVADR[0:3], address 0x4005 0048 (SLVADR0) to 0x4005 0054 (SLVADR3)) bit description

Bit	Symbol	Value	Description	Reset value
0	SADISABLE		Slave Address n Disable.	1
		0	Enabled. Slave Address n is enabled and will be recognized with any changes specified by the SLVQUAL0 register.	
		1	Ignored Slave Address n is ignored.	
7:1	SLVADR		Seven bit slave address that is compared to received addresses if enabled.	0
31:8	-		Reserved. Read value is undefined, only zero should be written.	NA

16.6.14 Slave address Qualifier 0 register

The SLVQUAL0 register can alter how Slave Address 0 is interpreted.

Table 187. Slave address Qualifier 0 register (SLVQUAL0, address 0x4005 0058) bit description

Bit	Symbol	Value	Description	Reset Value
0	QUALMODE0		Reserved. Read value is undefined, only zero should be written.	0
		0	The SLVQUAL0 field is used as a logical mask for matching address 0.	
		1	The SLVQUAL0 field is used to extend address 0 matching in a range of addresses.	
7:1	SLVQUAL0		Slave address Qualifier for address 0. A value of 0 causes the address in SLVADR0 to be used as-is, assuming that it is enabled. If QUALMODE0 = 0, any bit in this field which is set to 1 will cause an automatic match of the corresponding bit of the received address when it is compared to the SLVADR0 register. If QUALMODE0 = 1, an address range is matched for address 0. This range extends from the value defined by SLVADR0 to the address defined by SLVQUAL0 (address matches when SLVADR0[7:1] <= received address <= SLVQUAL0[7:1]).	0
31:8	-		Reserved. Read value is undefined, only zero should be written.	NA

16.6.15 Monitor data register

The read-only MONRXDAT register provides information about events on the I^2C bus, primarily to facilitate debugging of the I^2C during application development. All data addresses and data passing on the bus and whether these were acknowledged, as well as Start and Stop events, are reported.

The Monitor function must be enabled by the MONEN bit in the CFG register. Monitor mode can be configured to stretch the I^2C clock if data is not read from the MONRXDAT register in time to prevent it, via the MONCLKSTR bit in the CFG register**.** This can help ensure that nothing is missed but can cause the monitor function to be somewhat intrusive (by potentially adding clock delays, depending on software response time). In order to improve the chance of collecting all Monitor information if clock stretching is not enabled, Monitor data is buffered such that it is available until the end of the next piece of information from the I^2C bus.

Table 188. Monitor data register (MONRXDAT, address 0x4005 0080) bit description

Bit	Symbol	Value	Description	Reset value
7:0	MONRXDAT		Monitor function Receiver Data. This reflects every data byte that passes on the I^2C pins, and adds indication of Start, Repeated Start, and data NACK.	0
8	MONSTART		Monitor Received Start.	0
		0	No detect. The monitor function has not detected a Start event on the I^2C bus.	
		1	Start detect. The monitor function has detected a Start event on the I^2C bus.	

Table 188. Monitor data register (MONRXDAT, address 0x4005 0080) bit description

Bit	Symbol	Value	Description	Reset value
9	MONRESTART		Monitor Received Repeated Start.	0
		0	No start detect. The monitor function has not detected a Repeated Start event on the I^2C bus.	
		1	Repeated start detect. The monitor function has detected a Repeated Start event on the I^2C bus.	
10	MONNACK		Monitor Received NACK.	0
		0	Acknowledged. The data currently being provided by the monitor function was acknowledged by at least one master or slave receiver.	
		1	Not acknowledged. The data currently being provided by the monitor function was not acknowledged by any receiver.	
31:11	-		Reserved. Read value is undefined, only zero should be written.	NA

16.7 Functional description

16.7.1 Bus rates and timing considerations

Due to the nature of the I^2C bus, it is generally not possible to guarantee a specific clock rate on the SCL pin. On the I2C-bus, the clock can be stretched by any slave device, extended by software overhead time, etc. In a multi-master system, the master that provides the shortest SCL high time will cause that time to appear on SCL as long as that master is participating in I2C traffic (i.e. when it is the only master on the bus or during arbitration between masters).

Rate calculations give a base frequency that represents the fastest that the I^2C bus could operate if nothing slows it down.

16.7.1.1 Rate calculations

SCL high time (in I^2C function clocks) = (CLKDIV + 1) * (MSTSCLHIGH + 2)

SCL low time (in I^2C function clocks) = (CLKDIV + 1) * (MSTSCLLOW + 2)

Nominal SCL rate = I^2C function clock rate / (SCL high time + SCL low time)

16.7.2 Time-out

A time-out feature on an I^2C interface can be used to detect a “stuck” bus and potentially do something to alleviate the condition. Two different types of time-out are supported. Both types apply whenever the I^2C block and the time-out function are both enabled, Master, Slave, or Monitor functions do not need to be enabled.

In the first type of time-out, reflected by the EVENTTIMEOUT flag in the STAT register, the time between bus events governs the time-out check. These events include Start, Stop, and all changes on the I^2C clock (SCL). This time-out is asserted when the time between

any of these events is longer than the time configured in the TIMEOUT register. This time-out could be useful in monitoring an I^2C bus within a system as part of a method to keep the bus running of problems occur.

The second type of I^2C time-out is reflected by the SCLTIMEOUT flag in the STAT register. This time-out is asserted when the SCL signal remains low longer than the time configured in the TIMEOUT register. This corresponds to SMBus time-out parameter $T_{TIMEOUT}$. In this situation, a slave could reset its own I^2C interface in case it is the offending device. If all listening slaves (including masters that can be addressed as slaves) do this, then the bus will be released unless it is a current master causing the problem. Refer to the SMBus specification for more details.

Both types of time-out are generated when the I^2C bus is considered busy.

16.7.3 Ten-bit addressing

Ten-bit addressing is accomplished by the I^2C master sending a second address byte to extend a particular range of standard 7-bit addresses. In the case of the master writing to the slave, the I^2C frame simply continues with data after the 2 address bytes. For the master to read from a slave, it needs to reverse the data direction after the second address byte. This is done by sending a Repeated Start, followed by a repeat of the same standard 7-bit address, with a Read bit. The slave must remember that it had been addressed by the previous write operation and stay selected for the subsequent read with the correct partial I^2C address.

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

For the Slave function, the first part of the address is automatically matched in the same fashion as 7-bit addressing. The Slave address qualifier feature (see Section 16.6.14) can be used to intercept all potential 10-bit addresses (first address byte values F0 through F6), or just one. In the case of Slave Receiver mode, data is received in the normal fashion after software matches the first data byte to the remaining portion of the 10-bit address. The Slave function should record the fact that it has been addressed, in case there is a follow-up read operation.

For Slave Transmitter mode, the slave function responds to the initial address in the same fashion as for Slave Receiver mode, and checks that it has previously been addressed with a full 10-bit address. If the address matched is address 0, and address qualification is enabled, software must check that the first part of the 10-bit address is a complete match to the previous address before acknowledging the address.

16.7.4 Clocking and power considerations

The Master function of the I^2C always requires a peripheral clock to be running in order to operate. The Slave function can operate without any internal clocking when the slave is not currently addressed. This means that reduced power modes up to Power-down mode can be entered, and the device will wake up when the I^2C Slave function recognizes an address. Monitor mode can similarly wake up the device from a reduced power mode when information becomes available.

16.7.5 Interrupts

The I2C provides a single interrupt output that handles all interrupts for Master, Slave, and Monitor functions.

17.1 How to read this chapter

SPI0 is available on all parts. SPI1 is available on parts LPC812M101FDH16 and LPC812M101FDH20 only.

17.2 Features

- Data frames of 1 to 16 bits supported directly. Larger frames supported by software.
- Master and slave operation.
- Data can be transmitted to a slave without the need to read incoming data. This can be useful while setting up an SPI memory.
- Control information can optionally be written along with data. This allows very versatile operation, including “any length” frames.
- One Slave Select input/output with selectable polarity and flexible usage.

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

17.3 Basic configuration

Configure SPI0/1 using the following registers:

- In the SYSAHBCLKCTRL register, set bit 11 and 12 (Table 18) to enable the clock to the register interface.
- Clear the SPI0/1 peripheral resets using the PRESETCTRL register (Table 7).
- Enable/disable the SPI0/1 interrupts in interrupt slots #0 and 1 in the NVIC.
- Configure the SPI0/1 pin functions through the switch matrix. See Section 17.4.
- The peripheral clock for both SPIs is the system clock (see Figure 3 “LPC800 clock generation”).

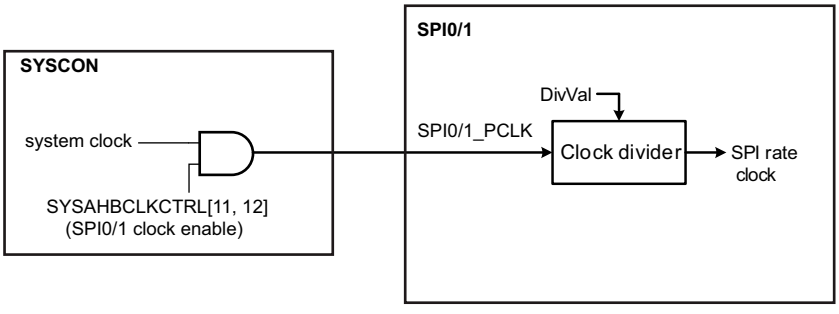


Fig 32. SPI clocking

17.3.1 Configure the SPIs for wake-up

In sleep mode, any signal that triggers an SPI interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the SPI clock SPI_PCLK remains active in sleep mode, the SPI can wake up the part independently of whether the SPI block is configured in master or slave mode.

In Deep-sleep or Power-down mode, the SPI clock is turned off as are all peripheral clocks. However, if the SPI is configured in slave mode and an external master provides the clock signal, the SPI can create an interrupt asynchronously. This interrupt, if enabled in the STARTERP1 register, in the NVIC, and in the SPI's INTENSET register, can then wake up the core.

17.3.1.1 Wake-up from Sleep mode

- Configure the SPI in either master or slave mode. See Table 191.
- Enable the SPI interrupt in the NVIC.
- Any SPI interrupt wakes up the part from sleep mode. Enable the SPI interrupt in the INTENSET register (Table 194).

17.3.1.2 Wake-up from Deep-sleep or Power-down mode

- Configure the SPI in slave mode. See Table 191. You must connect the SCK function to a pin and connect the pin to the master.
- Enable the SPI interrupt in the STARTERP1 register. See Table 34 "Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description".
- In the PDAWAKE register, configure all peripherals that need to be running when the part wakes up.
- Enable the SPI interrupt in the NVIC.
- Enable the interrupt in the INTENSET register which configures the interrupt as wake-up event (Table 194).. Examples are the following wake-up events:
  - A change in the state of the SSEL pin.
  - Data available to be received.
  - Receiver overrun.

17.4 Pin description

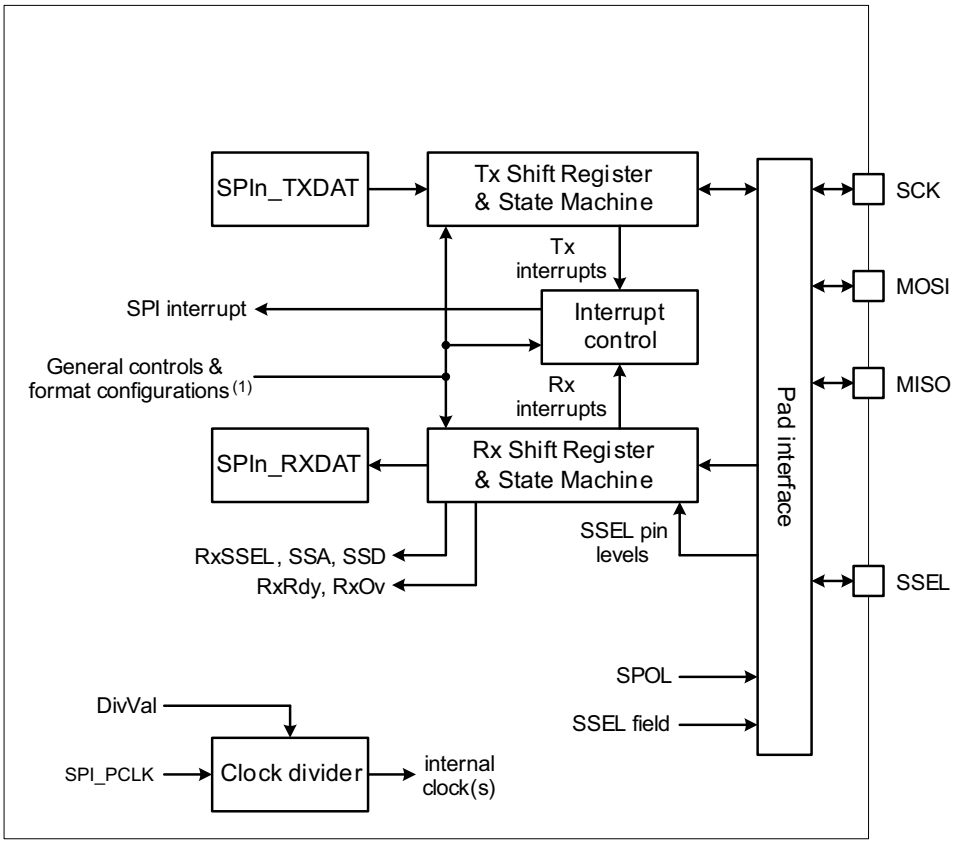
The SPI signals are movable functions and are assigned to external pins through the switch matrix.

See Section 9.3.1 "Connect an internal signal to a package pin" to assign the SPI functions to pins on the LPC800 package.

Table 189: SPI Pin Description

Function	Direction	Pin	Description	SWM register	Reference
SPI0_SCK	I/O	any	**Serial Clock.** SCK is a clock signal used to synchronize the transfer of data. It is driven by the master and received by the slave. When the SPI interface is used, the clock is programmable to be active-high or active-low. SCK only switches during a data transfer. It is driven whenever the Master bit in the CFG register equals 1, regardless of the state of the Enable bit.	PINASSIGN3	Table 100
SPI0_MOSI	I/O	any	**Master Out Slave In.** The MOSI signal transfers serial data from the master to the slave. When the SPI is a master, it outputs serial data on this signal. When the SPI is a slave, it clocks in serial data from this signal. MOSI is driven whenever the Master bit in the CFG register equals 1, regardless of the state of the Enable bit.	PINASSIGN4	Table 101
SPI0_MISO	I/O	any	**Master In Slave Out.** The MISO signal transfers serial data from the slave to the master. When the SPI is a master, serial data is input from this signal. When the SPI is a slave, serial data is output to this signal. MISO is driven when the SPI block is enabled, the Master bit in the CFG register equals 0, and when the slave is selected by one or more SSEL signals.	PINASSIGN4	Table 101
SPI0_SSEL	I/O	any	**Slave Select .** When the SPI interface is a master, it will drive the SSEL signals to an active state before the start of serial data and then release them to an inactive state after the serial data has been sent. By default, this signal is active low but can be selected to operate as active high. When the SPI is a slave, any SSEL in an active state indicates that this slave is being addressed. The SSEL pin is driven whenever the Master bit in the CFG register equals 1, regardless of the state of the Enable bit.	PINASSIGN4	Table 101
SPI1_SCK	I/O	any	Serial Clock.	PINASSIGN4	Table 101
SPI1_MOSI	I/O	any	Master Out Slave In.	PINASSIGN5	Table 102
SPI1_MISO	I/O	any	Master In Slave Out.	PINASSIGN5	Table 102
SPI1_SSEL	I/O	any	Slave Select.	PINASSIGN5	Table 102

17.5 General description



(1) Includes CPOL, CPHA, LSBF, FLEN, master, enable, transfer_delay, frame_delay, pre_delay, post_delay, SOT, EOT, EOF, RXIgnore, individual interrupt enables.

Fig 33. SPI block diagram

17.6 Register description

The Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

Table 190. Register overview: SPI (base address 0x4005 8000 (SPI0) and 0x4008 C000 (SPI1))

Name	Access	Offset	Description	Reset value	Reference
CFG	R/W	0x000	SPI Configuration register	0	Table 191
DLY	R/W	0x004	SPI Delay register	0	Table 192
STAT	R/W	0x008	SPI Status. Some status flags can be cleared by writing a 1 to that bit position	0x0102	Table 193

Table 190. Register overview: SPI (base address 0x4005 8000 (SPI0) and 0x4008 C000 (SPI1))
…continued

Name	Access	Offset	Description	Reset value	Reference
INTENSET	R/W	0x00C	SPI Interrupt Enable read and Set. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set.	0	Table 194
INTENCLR	W	0x010	SPI Interrupt Enable Clear. Writing a 1 to any implemented bit position causes the corresponding bit in INTENSET to be cleared.	NA	Table 195
RXDAT	R	0x014	SPI Receive Data	NA	Table 196
TXDATCTL	R/W	0x018	SPI Transmit Data with Control	0	Table 197
TXDAT	R/W	0x01C	SPI Transmit Data	0	Table 198
TXCTL	R/W	0x020	SPI Transmit Control	0	Table 199
DIV	R/W	0x024	SPI clock Divider	0	Table 200
INTSTAT	R	0x028	SPI Interrupt Status	0x02	Table 201

17.6.1 SPI Configuration register

The CFG register contains information for the general configuration of the SPI. Typically, this information is not changed during operation. Some configurations, such as CPOL, CPHA, and LSBF should not be made while the SPI is not fully idle. See the description of the Idle status (in Table 193) for more information.

Remark: If the interface is re-configured from Master mode to Slave mode or the reverse (an unusual case), the SPI should be disabled and re-enabled with the new configuration.

Table 191. SPI Configuration register (CFG, addresses 0x4005 8000 (SPI0) , 0x4005 C000 (SPI1)) bit description

Bit	Symbol	Value	Description	Reset value
0	Enable		SPI enable.	0
		0	Disabled. The SPI is disabled and the internal state machine and counters are reset.	
		1	Enabled. The SPI is enabled for operation.	
1	-		Reserved. Read value is undefined, only zero should be written.	NA
2	Master		Master mode select.	0
		0	Slave mode. The SPI will operate in slave mode. SCK, MOSI, and the SSEL signals are inputs, MISO is an output.	
		1	Master mode. The SPI will operate in master mode. SCK, MOSI, and the SSEL signals are outputs, MISO is an input.	
3	LSBF		LSB First mode enable.	0
		0	Standard. Data is transmitted and received in standard MSB first order.	
		1	Reverse. Data is transmitted and received in reverse order (LSB first).	
4	CPHA		Clock Phase select.	0
		0	Change. The SPI captures serial data on the first clock transition of the frame (when the clock changes away from the rest state). Data is changed on the following edge.	
		1	Capture. The SPI changes serial data on the first clock transition of the frame (when the clock changes away from the rest state). Data is captured on the following edge.	
5	CPOL		Clock Polarity select.	0
		0	Low. The rest state of the clock (between frames) is low.	
		1	High. The rest state of the clock (between frames) is high.	
6	-		Reserved. Read value is undefined, only zero should be written.	NA
7	LOOP		Loopback mode enable. Loopback mode applies only to Master mode, and connects transmit and receive data connected together to allow simple software testing.	0
		0	Disabled.	
		1	Enabled.	
8	SPOL		SSEL Polarity select.	0
		0	Low. The SSEL pin is active low. The value in the SSEL fields of the RXDAT, TXDATCTL, and TXCTL registers related to SSEL is not inverted relative to the pins.	
		1	High. The SSEL pin is active high. The value in the SSEL fields of the RXDAT, TXDATCTL, and TXCTL registers related to SSEL is inverted relative to the pins.	
31:9	-		Reserved. Read value is undefined, only zero should be written.	NA

17.6.2 SPI Delay register

The DLY register controls several programmable delays related to SPI signalling. These delays apply only to master mode, and are all stated in SPI clocks.

Timing details are shown in:

Section 17.7.2.1 “Pre_delay and Post_delay”

Section 17.7.2.2 “Frame_delay”

Section 17.7.2.3 “Transfer_delay”

Table 192. SPI Delay register (DLY, addresses 0x4005 8004 (SPI0) , 0x4005 C004 (SPI1)) bit description

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

17.6.3 SPI Status register

The STAT register provides SPI status flags for software to read, and a control bit for forcing an end of transfer. Flags other than read-only flags may be cleared by writing ones to corresponding bits of STAT.

STAT contains 2 error flags (in slave mode only): RXOV and TXUR. These are receiver overrun and transmit underrun, respectively. If either of these errors occur during operation, the SPI should be disabled, then re-enabled in order to make sure all internal states are cleared before attempting to resume operation.

In this register, the following notation is used: RO = Read-only, W1 = write 1 to clear..

Table 193. SPI Status register (STAT, addresses 0x4005 8008 (SPI0) , 0x4005 C008 (SPI1)) bit description

Bit	Symbol	Description	Reset value	Access [1]
0	RXRDY	Receiver Ready flag. When 1, indicates that data is available to be read from the receiver buffer. Cleared after a read of the RXDAT register.	0	RO
1	TXRDY	Transmitter Ready flag. When 1, this bit indicates that data may be written to the transmit buffer. Previous data may still be in the process of being transmitted. Cleared when data is written to TXDAT or TXDATCTL until the data is moved to the transmit shift register.	1	RO
2	RXOV	Receiver Overrun interrupt flag. This flag applies only to slave mode (Master = 0). This flag is set when the beginning of a received character is detected while the receiver buffer is still in use. If this occurs, the receiver buffer contents are preserved, and the incoming data is lost. Data received by the SPI should be considered undefined if RxOv is set.	0	W1
3	TXUR	Transmitter Underrun interrupt flag. This flag applies only to slave mode (Master = 0). In this case, the transmitter must begin sending new data on the next input clock if the transmitter is idle. If that data is not available in the transmitter holding register at that point, there is no data to transmit and the TXUR flag is set. Data transmitted by the SPI should be considered undefined if TXUR is set.	0	W1
4	SSA	Slave Select Assert. This flag is set whenever any slave select transitions from deasserted to asserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become busy, and allows waking up the device from reduced power modes when a slave mode access begins. This flag is cleared by software.	0	W1
5	SSD	Slave Select Deassert. This flag is set whenever any asserted slave selects transition to deasserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become idle. This flag is cleared by software.	0	W1
6	STALLED	Stalled status flag. This indicates whether the SPI is currently in a stall condition.	0	RO

Table 193. SPI Status register (STAT, addresses 0x4005 8008 (SPI0) , 0x4005 C008 (SPI1)) bit description

Bit	Symbol	Description	Reset value	Access [1]
7	ENDTRANSFER	End Transfer control bit. Software can set this bit to force an end to the current transfer when the transmitter finishes any activity already in progress, as if the EOT flag had been set prior to the last transmission. This capability is included to support cases where it is not known when transmit data is written that it will be the end of a transfer. The bit is cleared when the transmitter becomes Idle as the transfer comes to an end. Forcing an end of transfer in this manner causes any specified FrameDelay and TransferDelay to be inserted.	0	RO/W1
8	IDLE	Idle status flag. This bit is 1 whenever the SPI master function is fully idle. This means that the transmit holding register is empty and the transmitter is not in the process of sending data.	1	RO
31:9	-	Reserved. Read value is undefined, only zero should be written.	NA	NA

[1] RO = Read-only, W1 = write 1 to clear.

17.6.4 SPI Interrupt Enable read and Set register

The INTENSET register is used to enable various SPI interrupt sources. Enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The INTENCLR register is used to clear bits in this register. See Table 193 for details of the interrupts.

Table 194. SPI Interrupt Enable read and Set register (INTENSET, addresses 0x4005 800C (SPI0) , 0x4005 C00C (SPI1)) bit description

Bit	Symbol	Value	Description	Reset value
0	RXRDYEN		Determines whether an interrupt occurs when receiver data is available.	0
		0	No interrupt will be generated when receiver data is available.	
		1	An interrupt will be generated when receiver data is available in the RXDAT register.	
1	TXRDYEN		Determines whether an interrupt occurs when the transmitter holding register is available.	0
		0	No interrupt will be generated when the transmitter holding register is available.	
		1	An interrupt will be generated when data may be written to TXDAT.	
2	RXOVEN		Determines whether an interrupt occurs when a receiver overrun occurs. This happens in slave mode when there is a need for the receiver to move newly received data to the RXDAT register when it is already in use. The interface prevents receiver overrun in Master mode by not allowing a new transmission to begin when a receiver overrun would otherwise occur.	0
		0	No interrupt will be generated when a receiver overrun occurs.	
		1	An interrupt will be generated if a receiver overrun occurs.	
3	TXUREN		Determines whether an interrupt occurs when a transmitter underrun occurs. This happens in slave mode when there is a need to transmit data when none is available.	0
		0	No interrupt will be generated when the transmitter underruns.	
		1	An interrupt will be generated if the transmitter underruns.	

Table 194. SPI Interrupt Enable read and Set register (INTENSET, addresses 0x4005 800C (SPI0) , 0x4005 C00C (SPI1)) bit description

Bit	Symbol	Value	Description	Reset value
4	SSAEN		Determines whether an interrupt occurs when the Slave Select is asserted.	0
		0	No interrupt will be generated when any Slave Select transitions from deasserted to asserted.	
		1	An interrupt will be generated when any Slave Select transitions from deasserted to asserted.	
5	SSDEN		Determines whether an interrupt occurs when the Slave Select is deasserted.	0
		0	No interrupt will be generated when all asserted Slave Selects transition to deasserted.	
		1	An interrupt will be generated when all asserted Slave Selects transition to deasserted.	
31:6	-		Reserved. Read value is undefined, only zero should be written.	NA

17.6.5 SPI Interrupt Enable Clear register

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

Table 195. SPI Interrupt Enable clear register (INTENCLR, addresses 0x4005 8010 (SPI0) , 0x4005 C010 (SPI1)) bit description

Bit	Symbol	Description	Reset value
0	RXRDYEN	Writing 1 clears the corresponding bits in the INTENSET register.	0
1	TXRDYEN	Writing 1 clears the corresponding bits in the INTENSET register.	0
2	RXOVEN	Writing 1 clears the corresponding bits in the INTENSET register.	0
3	TXUREN	Writing 1 clears the corresponding bits in the INTENSET register.	0
4	SSAEN	Writing 1 clears the corresponding bits in the INTENSET register.	0
5	SSDEN	Writing 1 clears the corresponding bits in the INTENSET register.	0
31:6	-	Reserved. Read value is undefined, only zero should be written.	NA

17.6.6 SPI Receiver Data register

The read-only RXDAT register provides the means to read the most recently received data. The value of SSEL can be read along with the data.

For details on the slave select process, see Section 17.7.4.

Table 196. SPI Receiver Data register (RXDAT, addresses 0x4005 8014 (SPI0) , 0x4005 C014 (SPI1)) bit description

Bit	Symbol	Description	Reset value
15:0	RXDAT	Receiver Data. This contains the next piece of received data. The number of bits that are used depends on the FLen setting in TXCTL / TXDATCTL.	undefined
16	RXSSELN	Slave Select for receive. This field allows the state of the SSEL pin to be saved along with received data. The value will reflect the SSEL pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	undefined
19:17	-	Reserved.	-
20	SOT	Start of Transfer flag. This flag will be 1 if this is the first frame after SSEL went from deasserted to asserted (i.e., any previous transfer has ended). This information can be used to identify the first piece of data in cases where the frame length is greater than 16 bit.	
31:21	-	Reserved, the value read from a reserved bit is not defined.	NA

17.6.7 SPI Transmitter Data and Control register

The TXDATCTL register provides a location where both transmit data and control information can be written simultaneously. This allows detailed control of the SPI without a separate write of control information for each piece of data.

When control information remains static during transmit, the TXDAT register should be used (see Section 17.6.8) instead of the TXDATCTL register. Control information can then be written separately via the TXCTL register (see Section 17.6.9). The upper part of TXDATCTL (bits 27 to 16) are the same bits contained in the TXCTL register. The two registers simply provide two ways to access them.

For details on the slave select process, see Section 17.7.4.

For details on using multiple consecutive frames for frame lengths larger than 16 bit, see Section 17.7.5 “Data lengths greater than 16 bits”.

Table 197. SPI Transmitter Data and Control register (TXDATCTL, addresses 0x4005 8018 (SPI0) , 0x4005 C018 (SPI1)) bit description

Bit	Symbol	Value	Description	Reset value
15:0	TXDAT		Transmit Data. This field provides from 1 to 16 bits of data to be transmitted.	0
16	TXSSELN		Transmit Slave Select . This field controls what is output for SSEL in master mode. **Remark:** The active state of the SSEL function is configured by bits in the CFG register.	0
		0	SSEL asserted.	
		1	SSEL not asserted.	
19:17	-		Reserved.	
20	EOT		End of Transfer. The asserted SSEL will be deasserted at the end of a transfer, and remain so for at least the time specified by the Transfer_delay value in the DLY register.	0
		0	SSEL not deasserted. This piece of data is not treated as the end of a transfer. SSEL will not be deasserted at the end of this data.	
		1	SSEL deasserted. This piece of data is treated as the end of a transfer. SSEL will be deasserted at the end of this piece of data.	
21	EOF		End of Frame. Between frames, a delay may be inserted, as defined by the Frame_delay value in the DLY register. The end of a frame may not be particularly meaningful if the FRAME_DELAY value = 0. This control can be used as part of the support for frame lengths greater than 16 bits.	0
		0	Data not EOF. This piece of data transmitted is not treated as the end of a frame.	
		1	Data EOF. This piece of data is treated as the end of a frame, causing the FRAME_DELAY time to be inserted before subsequent data is transmitted.	
22	RXIGNORE		Receive Ignore. This allows data to be transmitted using the SPI without the need to read unneeded data from the receiver to simplify the transmit process.	0
		0	Read received data. Received data must be read in order to allow transmission to progress. In slave mode, an overrun error will occur if received data is not read before new data is received.	
		1	Ignore received data. Received data is ignored, allowing transmission without reading unneeded received data. No receiver flags are generated.	

Table 197. SPI Transmitter Data and Control register (TXDATCTL, addresses 0x4005 8018 (SPI0) , 0x4005 C018 (SPI1)) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
23	-		Reserved. Read value is undefined, only zero should be written.	NA
27:24	FLEN		Frame Length. Specifies the frame length from 1 to 16 bits. Note that frame lengths greater than 16 bits are supported by implementing multiple sequential frames. Note that if a 1-bit frame is selected, the master function will always insert a delay with a length of one SCK time following the single clock seen on the SCK pin. 0x0 = Data frame is 1 bit in length. 0x1 = Data frame is 2 bits in length. 0x2 = Data frame is 3 bits in length. ... 0xF = Data frame is 16 bits in length.	0x0
31:28	-		Reserved. Read value is undefined, only zero should be written.	NA

17.6.8 SPI Transmitter Data Register

The TXDAT register is written in order to send data via the SPI transmitter when control information is not changing during the transfer (see Section 17.6.7). That data will be sent to the transmit shift register when it is available, and another character may then be written to TXDAT.

Table 198. SPI Transmitter Data Register (TXDAT, addresses 0x4005 801C (SPI0) , 0x4005 C01C (SPI1)) bit description

Bit	Symbol	Description	Reset value
15:0	DATA	Transmit Data. This field provides from 4 to 16 bits of data to be transmitted.	0
31:16	-	Reserved. Only zero should be written.	NA

17.6.9 SPI Transmitter Control register

The TXCTL register provides a way to separately access control information for the SPI. These bits are another view of the same-named bits in the TXDATCTL register (see Section 17.6.7). Changing bits in TXCTL has no effect unless data is later written to the TXDAT register. Data written to TXDATCTL overwrites the TXCTL register.

When control information needs to be changed during transmission, the TXDATCTL register should be used (see Section 17.6.7) instead of TXDAT. Control information can then be written along with data.

Table 199. SPI Transmitter Control register (TXCTL, addresses 0x4005 8020 (SPI0) , 0x4005 C020 (SPI1)) bit description

Bit	Symbol	Description	Reset value
15:0	-	Reserved. Read value is undefined, only zero should be written.	NA
16	TX SSEL	Transmit Slave Select.	0x0
19:17	-	Reserved.	0x0
20	EOT	End of Transfer.	0
21	EOF	End of Frame.	0
22	RXIGNORE	Receive Ignore.	0
23	-	Reserved. Read value is undefined, only zero should be written.	NA
27:24	FLEN	Frame Length.	0x0
31:28	-	Reserved. Read value is undefined, only zero should be written.	NA

17.6.10 SPI Divider register

The DIV register determines the clock used by the SPI in master mode.

For details on clocking, see Section 17.7.3 “Clocking and data rates”.

Table 200. SPI Divider register (DIV, addresses 0x4005 8024 (SPI0) , 0x4005 C024 (SPI1)) bit description

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

17.6.11 SPI Interrupt Status register

The read-only INTSTAT register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See Table 193 for detailed descriptions of the interrupt flags.

Table 201. SPI Interrupt Status register (INTSTAT, addresses 0x4005 8028 (SPI0) , 0x4005 C028 (SPI1)) bit description

Bit	Symbol	Description	Reset value
0	RXRDY	Receiver Ready flag.	0
1	TXRDY	Transmitter Ready flag.	1
2	RXOV	Receiver Overrun interrupt flag.	0
3	TXUR	Transmitter Underrun interrupt flag.	0

Table 201. SPI Interrupt Status register (INTSTAT, addresses 0x4005 8028 (SPI0) , 0x4005 C028 (SPI1)) bit description

Bit	Symbol	Description	Reset value
4	SSA	Slave Select Assert.	0
5	SSD	Slave Select Deassert.	0
31:6	-	Reserved. Read value is undefined, only zero should be written.	NA

17.7 Functional description

17.7.1 Operating modes: clock and phase selection

SPI interfaces typically allow configuration of clock phase and polarity. These are sometimes referred to as numbered SPI modes, as described in Table 202 and shown in Figure 34. CPOL and CPHA are configured by bits in the CFG register (Section 17.6.1).

Table 202: SPI mode summary

CPOL	CPHA	SPI Mode	Description	SCK rest state	SCK data change edge	SCK data sample edge
0	0	0	The SPI captures serial data on the first clock transition of the frame (when the clock changes away from the rest state). Data is changed on the following edge.	low	falling	rising
0	1	1	The SPI changes serial data on the first clock transition of the frame (when the clock changes away from the rest state). Data is captured on the following edge.	low	rising	falling
1	0	2	Same as mode 0 with SCK inverted.	high	rising	falling
1	1	3	Same as mode 1 with SCK inverted.	high	falling	rising

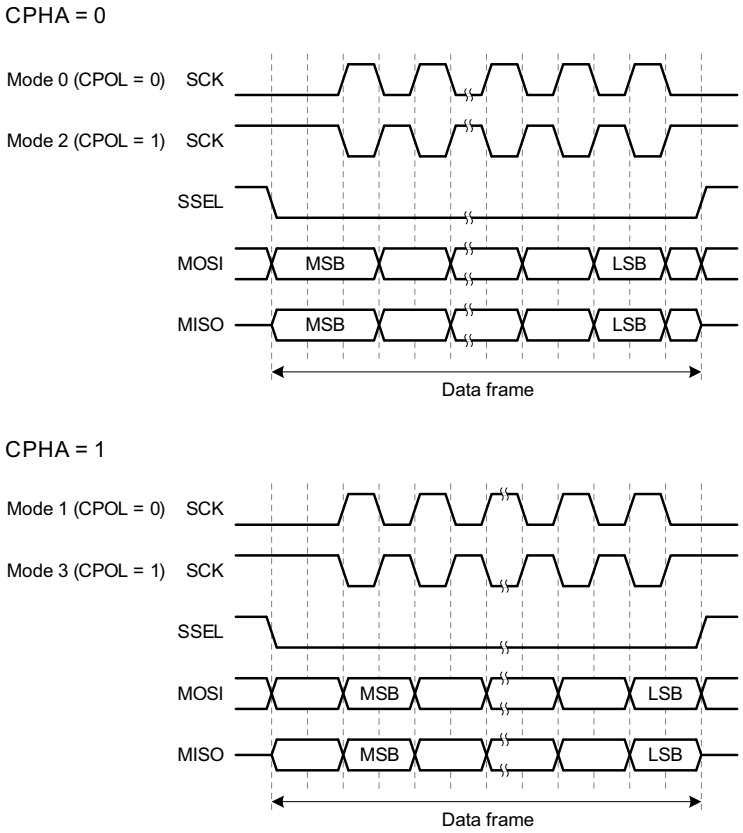


Fig 34. Basic SPI operating modes

17.7.2 Frame delays

Several delays can be specified for SPI frames. These include:

- Pre_delay: delay after SSEL is asserted before data clocking begins
- Post_delay: delay at the end of a data frame before SSEL is deasserted
- Frame_delay: delay between data frames when SSEL is not deasserted
- Transfer_delay: minimum duration of SSEL in the deasserted state between transfers

17.7.2.1 Pre_delay and Post_delay

Pre_delay and Post_delay are illustrated by the examples in Figure 35. The Pre_delay value controls the amount of time between SSEL being asserted and the beginning of the subsequent data frame. The Post_delay value controls the amount of time between the end of a data frame and the deassertion of SSEL.

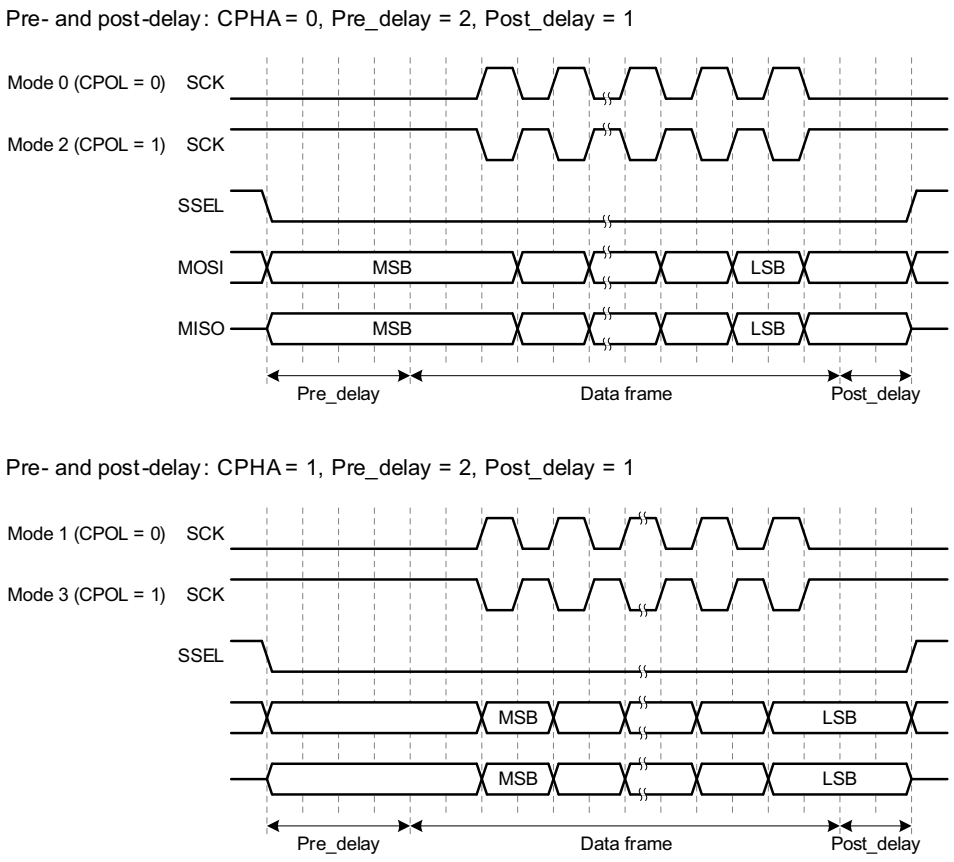


Fig 35. Pre_delay and Post_delay timing

17.7.2.2 Frame_delay

The Frame_delay value controls the amount of time at the end of each frame. This delay is inserted when the EOF bit = 1. Frame_delay is illustrated by the examples in Figure 36. Note that frame boundaries occur only where specified. This is because frame lengths can be any size, involving multiple data writes. See Section 17.7.5 for more information.

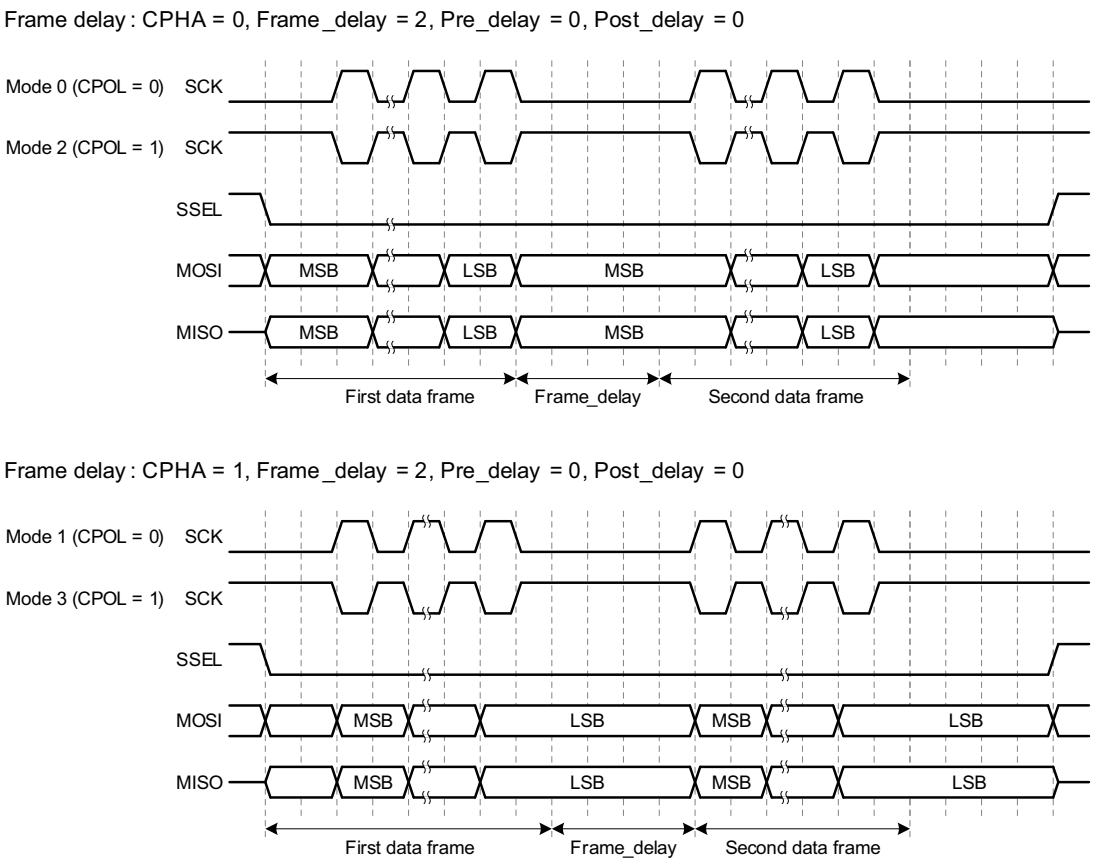


Fig 36. Frame_delay timing

17.7.2.3 Transfer_delay

The Transfer_delay value controls the minimum amount of time that SSEL is deasserted between transfers, because the EOT bit = 1. When Transfer_delay = 0, SSEL may be deasserted for a minimum of one SPI clock time. Transfer_delay is illustrated by the examples in Figure 37.

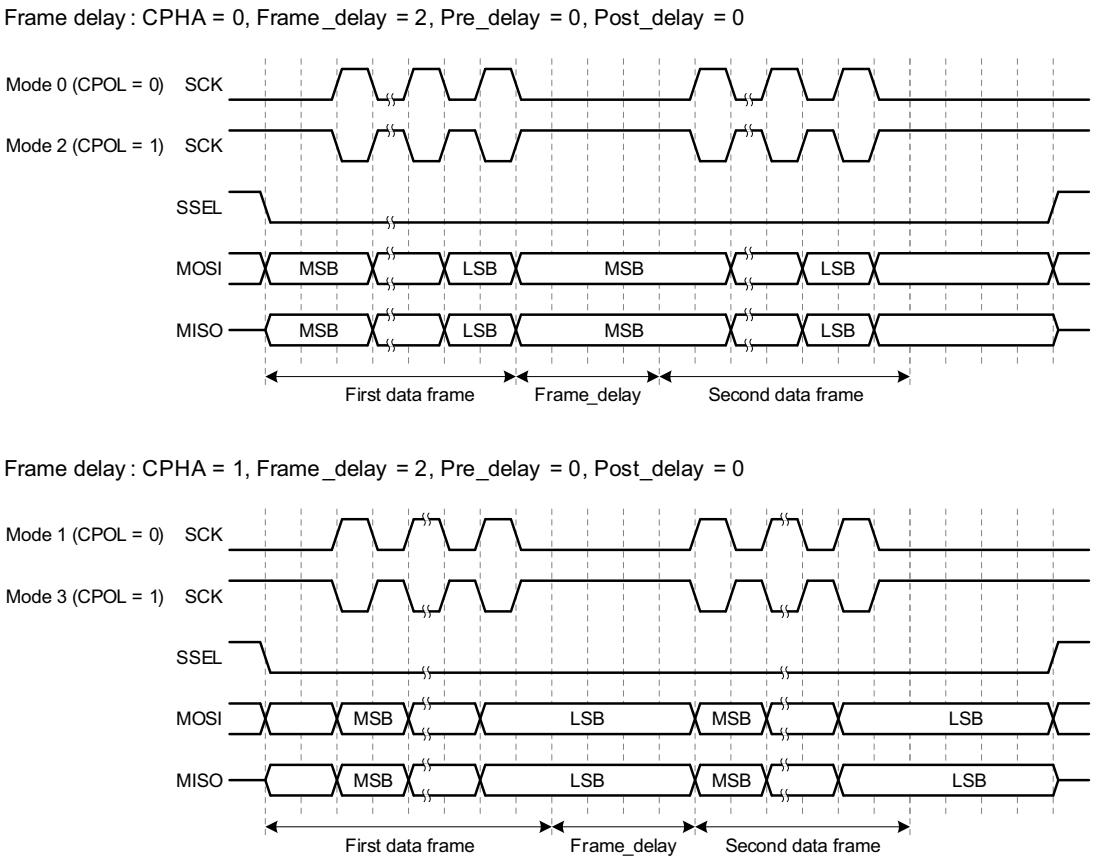


Fig 37. Transfer_delay timing

17.7.3 Clocking and data rates

In order to use the SPI, clocking details must be defined. This includes configuring the system clock and selection of the clock divider value in DIV. See Figure 32.

17.7.3.1 Data rate calculations

The SPI interface is designed to operate asynchronously from any on-chip clocks, and without the need for overclocking.

In slave mode, this means that the SCK from the external master is used directly to run the transmit and receive shift registers and other logic.

In master mode, the SPI rate clock produced by the SPI clock divider is used directly as the outgoing SCK.

The SPI clock divider is an integer divider. The SPI in master mode can be set to run at the same speed as the selected PCLK, or at lower integer divide rates. The SPI rate will be = PCLK_SPIn / DIVVAL.

In slave mode, the clock is taken from the SCK input and the SPI clock divider is not used.

17.7.4 Slave select

The SPI block provides for one Slave Select input in slave mode or output in master mode. The SSEL can be set for normal polarity (active low), or can be inverted (active high). Representation of the SSEL in a register is always active low. If the SSEL is inverted, this is done as the signal leaves/enters the SPI block.

In slave mode, the asserted SSEL that is connected to a pin will activate the SPI. In master mode, the SSEL that is connected to a pin will be output as defined in the SPI registers.

In master mode, the Slave Select is configured by the TXSSELN field, which appears in both the TXCTL and TXDATCTL registers. In slave mode, the state of the SSEL is saved along with received data in the RXSSELN field of the RXDAT register.

17.7.5 Data lengths greater than 16 bits

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

Details of how to handle larger data widths depend somewhat on other SPI configuration options. For instance, if it is intended for Slave Selects to be deasserted between frames, then this must be suppressed when a larger frame is split into more than one part. Sending 2 groups of 12 bits with SSEL deasserted between 24-bit increments, for instance, would require changing the value of the EOF bit on alternate 12-bit frames.

17.7.6 Data stalls

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

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

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

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

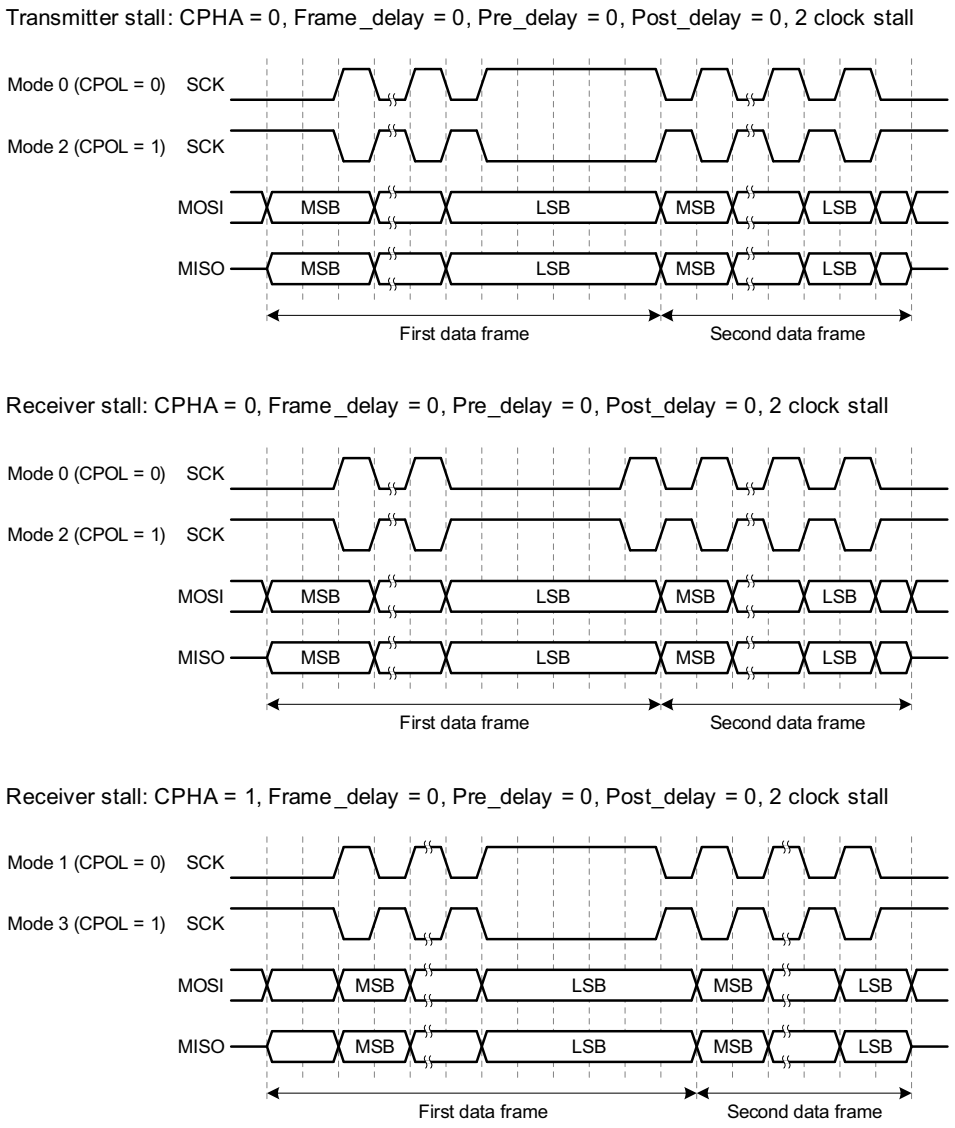


Fig 38. Examples of data stalls

18.1 How to read this chapter

The analog comparator is available on all LPC800 parts.

18.2 Features

- Selectable external inputs can be used as either the positive or negative input of the comparator.
- The Internal voltage reference (0.9 V bandgap reference) can be used as either the positive or negative input of the comparator.
- 32-stage voltage ladder can be used as either the positive or negative input of the comparator.
- Voltage ladder source selectable between the supply pin V_{DD} or VDDCMP pin.
- Voltage ladder can be separately powered down when not required.
- Interrupt capability

18.3 Basic configuration

Configure the analog comparator using the following registers:

- In the SYSAHBCLKCTRL register, set bit 19 (Table 18) to enable the clock to the register interface.
- You can enable or disable the power to the analog comparator through the PDRUNCFG register (Table 37).
- Clear the analog comparator peripheral reset using the PRESETCTRL register (Table 7).
- The analog comparator interrupt is connected to interrupt #11 in the NVIC.
- Configure the analog comparator pin functions through the switch matrix. See Section 18.4.

18.3.1 Connect the comparator output to the SCT

You can use the comparator output function (ACMP_O) to start or stop the SCT or, more generally, create an SCT event. To create an SCT event, connect AMP_O as follows:

1. Using the switch matrix, connect ACMP_O to a pin. See Table 203.
2. Using the switch matrix, connect any of the SCT input functions to the same pin. See Table 107.

The selected SCT input can now monitor the ACMP_O function.

18.4 Pin description

The analog comparator reference voltage, the inputs, and the output are assigned to external pins through the switch matrix. You can assign the analog comparator output to any pin on the package that is not a supply or ground pin. The comparator inputs and the reference voltage are fixed-pin functions that must be enabled through the switch matrix and can only be assigned to special pins on the package.

See Section 9.3.1 “Connect an internal signal to a package pin” to assign the analog comparator output to any pin on the LPC800 package.

See Section 9.3.2 to enable the analog comparator inputs and the reference voltage input.

Table 203. Analog comparator pin description

Function	Type	Pin	Description	SWM register	Reference
ACMP_I1	I	PIO0_0/ACMP_I1	Comparator input 1	PINENABLE0	Section 9.5.10
ACMP_I2	I	PIO0_0/ACMP_I2/CLKIN	Comparator input 2. Disable the CLKIN function in the PINENABLE0 register.	PINENABLE0	Section 9.5.10
ACMP_O	O	any	Comparator output	PINASSIGN8	Section 9.5.9
VDDCMP	I	PIO0_6/VDDCMP	External reference voltage source for 32-stage Voltage Ladder.	PINENABLE0	Section 9.5.10

18.5 General description

The analog comparator can compare voltage levels on external pins and internal voltages.

The comparator has 4 inputs multiplexed separately to its positive and negative inputs. The multiplexers are controlled by the comparator register CTL (see Figure 39 and Table 205).

Input 0 of the multiplexer is the programmable voltage ladder output.

Bits 2:1 control the external inputs ACMP_I[2:1].

Bits 6 of the multiplexer controls internal reference voltage input.

All other bits are reserved.

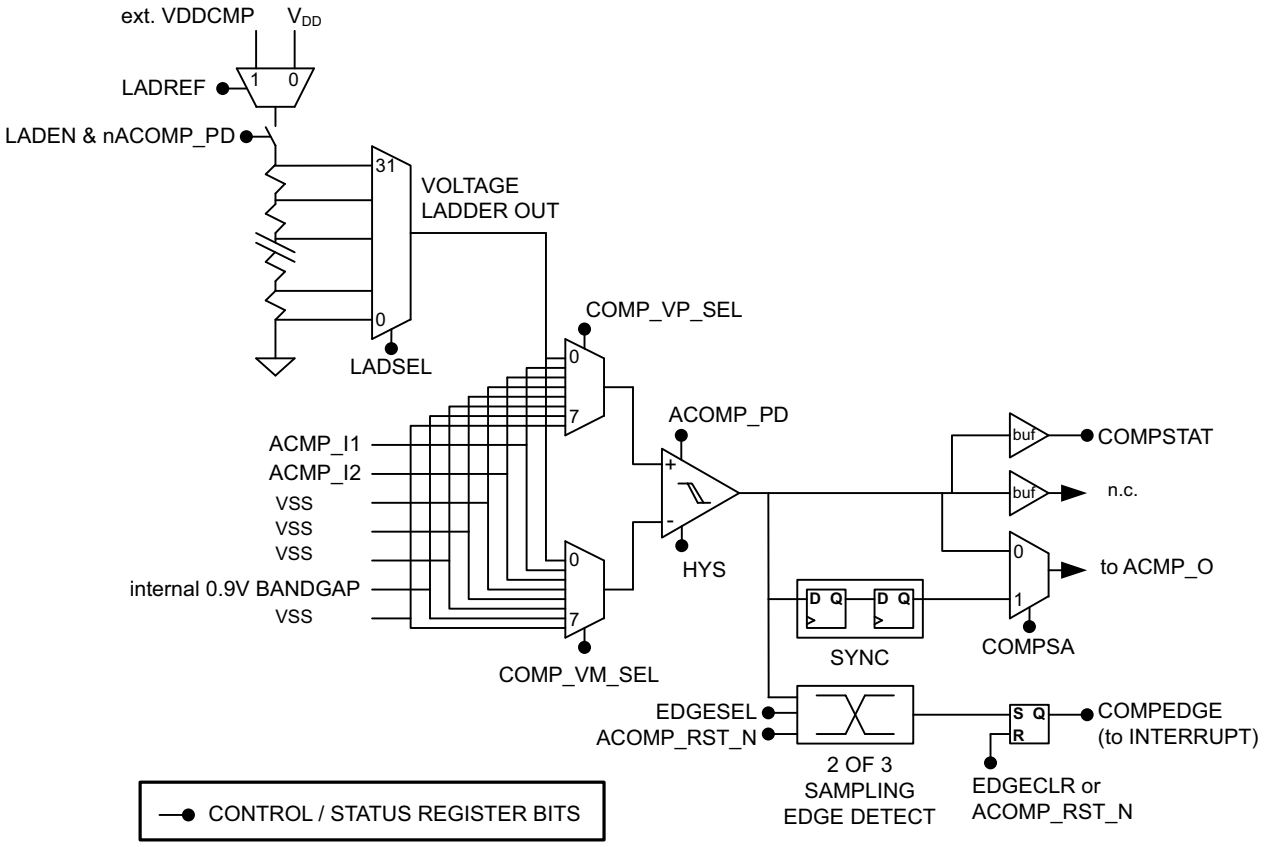


Fig 39. Comparator block diagram

18.5.1 Reference voltages

The voltage ladder can use two reference voltages, from the VDDCMP or the V_{DD} pin. The voltage ladder selects one of 32 steps between the pin voltage and V_{SS} inclusive. The voltage on VDDCMP should not exceed that on V_{DD} .

18.5.2 Settling times

After the voltage ladder is powered on, it requires stabilization time until comparisons using it are accurate. Much shorter settling times apply after the LADSEL value is changed and when either or both voltage sources are changed. Software can deal with these factors by repeatedly reading the comparator output until a number of readings yield the same result.

18.5.3 Interrupts

The interrupt output comes from edge detection circuitry in this module. Rising edges, falling edges, or both edges can set the COMPEDGE bit and thus request an interrupt. COMPEDGE and the interrupt request are cleared when software writes a 1 to EDGECLR.

18.5.4 Comparator outputs

The comparator output (conditioned by COMPSA bit) can be routed to an external pin. When COMPSA is 0 and the comparator interrupt is disabled, the comparator can be used with the bus clock disabled (Table 18 “System clock control register (SYSAHBCLKCTRL, address 0x4004 8080) bit description”) to save power if the control registers don’t need to be written.

The status of the comparator output can be observed through the comparator status register bit.

The comparator output can be routed to the SCT via the switch matrix allowing to capture the time of a voltage crossing or to count crossings in either or both directions. See Section 18.3.1 “Connect the comparator output to the SCT”.

18.6 Register description

Table 204. Register overview: Analog comparator (base address 0x4002 4000)

Name	Access	Address offset	Description	Reset value
CTRL	R/W	0x000	Comparator control register	0
LAD	R/W	0x004	Voltage ladder register	0

18.6.1 Comparator control register

This register enables the comparator, configures the interrupts, and controls the input multiplexers on both sides of the comparator. All bits not shown in Table 205 are reserved and should be written as 0.

Table 205. Comparator control register (CTRL, address 0x4002 4000) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved. Write as 0.	0
4:3	EDGESEL		This field controls which edges on the comparator output set the COMPEDGE bit (bit 23 below):	0
		0x0	Falling edges	
		0x1	Rising edges	
		0x2	Both edges	
		0x3	Both edges	
5	-		Reserved. Write as 0.	0
6	COMPSA		Comparator output control	0
		0	Comparator output is used directly.	
		1	Comparator output is synchronized to the bus clock for output to other modules.	
7	-		Reserved. Write as 0.	0

Table 205. Comparator control register (CTRL, address 0x4002 4000) bit description

Bit	Symbol	Value	Description	Reset value
10:8	COMP_VP_SEL		Selects positive voltage input	0
		0x0	Voltage ladder output	
		0x1	ACMP_I1	
		0x2	ACMP_I2	
		0x3	Reserved	
		0x4	Reserved	
		0x5	Reserved	
		0x6	Internal reference voltage (bandgap)	
		0x7	Reserved	
13:11	COMP_VM_SEL		Selects negative voltage input	0
		0x0	Voltage ladder output	
		0x1	ACMP_I1	
		0x2	ACMP_I2	
		0x3	Reserved	
		0x4	Reserved	
		0x5	Reserved	
		0x6	Internal reference voltage (bandgap)	
		0x7	Reserved	
19:14	-		Reserved. Write as 0.	0
20	EDGECLR		Interrupt clear bit. To clear the COMPEDGE bit and thus negate the interrupt request, toggle the EDGECLR bit by first writing a 1 and then a 0.	0
21	COMPSTAT		Comparator status. This bit reflects the state of the comparator output.	0
22	-		Reserved. Write as 0.	0
23	COMPEDGE		Comparator edge-detect status.	0
24	-		Reserved. Write as 0.	0
26:25	HYS		Controls the hysteresis of the comparator. When the comparator is outputting a certain state, this is the difference between the selected signals, in the opposite direction from the state being output, that will switch the output.	0
		0x0	None (the output will switch as the voltages cross)	
		0x1	5 mV	
		0x2	10 mV	
		0x3	20 mV	
31:27	-		Reserved	-

18.6.2 Voltage ladder register

This register enables and controls the voltage ladder. The fraction of the reference voltage produced by the ladder is programmable in steps of 1/31.

Table 206. Voltage ladder register (LAD, address 0x4002 4004) bit description

Bit	Symbol	Value	Description	Reset value
0	LADEN		Voltage ladder enable	0
5:1	LADSEL		Voltage ladder value. The reference voltage Vref depends on the LADREF bit below. 00000 = V_{SS} 00001 = 1 × Vref/31 00010 = 2 × Vref/31 ... 11111 = Vref	0
6	LADREF		Selects the reference voltage Vref for the voltage ladder:	0
		0	Supply pin V_{DD}	
		1	VDDCMP pin	
31:7	-		Reserved.	0

19.1 How to read this chapter

The CRC engine is available on all LPC800 parts.

19.2 Features

- Supports three common polynomials CRC-CCITT, CRC-16, and CRC-32.
  - CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$
  - CRC-16: $x^{16} + x^{15} + x^2 + 1$
  - CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Bit order reverse and 1’s complement programmable setting for input data and CRC sum.
- Programmable seed number setting.
- Accept any size of data width per write: 8, 16 or 32-bit.
  - 8-bit write: 1-cycle operation
  - 16-bit write: 2-cycle operation (8-bit x 2-cycle)
  - 32-bit write: 4-cycle operation (8-bit x 4-cycle)

19.3 Basic configuration

Enable the clock to the CRC engine in the SYSAHBCLKCTRL register (Table 18, bit 13).

19.4 Pin description

The CRC engine has no configurable pins.

19.5 General description

The Cyclic Redundancy Check (CRC) generator with programmable polynomial settings supports several CRC standards commonly used.

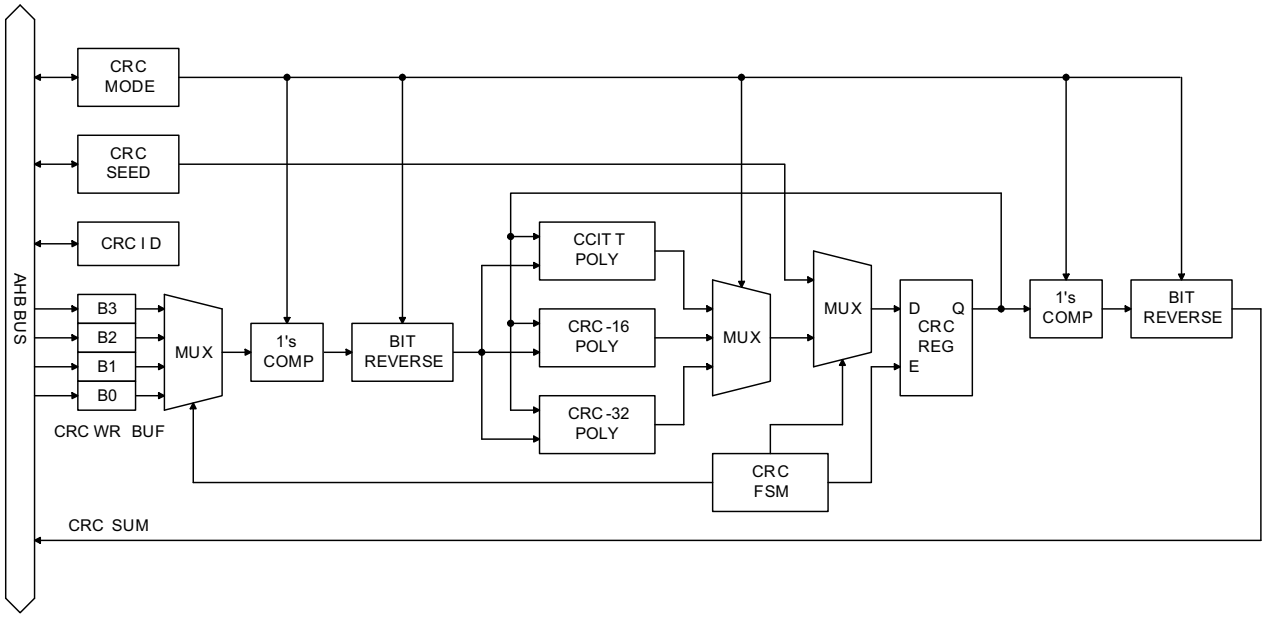


Fig 40. CRC block diagram

19.6 Register description

Table 207. Register overview: CRC engine (base address 0x5000 0000)

Name	Access	Address offset	Description	Reset value	Reference
MODE	R/W	0x000	CRC mode register	0x0000 0000	Table 208
SEED	R/W	0x004	CRC seed register	0x0000 FFFF	Table 209
SUM	RO	0x008	CRC checksum register	0x0000 FFFF	Table 210
WR_DATA	WO	0x008	CRC data register	-	Table 211

19.6.1 CRC mode register

Table 208. CRC mode register (MODE, address 0x5000 0000) bit description

Bit	Symbol	Description	Reset value
1:0	CRC_POLY	CRC polynom: 1X= CRC-32 polynomial 01= CRC-16 polynomial 00= CRC-CCITT polynomial	00
2	BIT_RVS_WR	Data bit order: 1= Bit order reverse for CRC_WR_DATA (per byte) 0= No bit order reverse for CRC_WR_DATA (per byte)	0
3	CMPL_WR	Data complement: 1= 1's complement for CRC_WR_DATA 0= No 1's complement for CRC_WR_DATA	0
4	BIT_RVS_SUM	CRC sum bit order: 1= Bit order reverse for CRC_SUM 0= No bit order reverse for CRC_SUM	0
5	CMPL_SUM	CRC sum complement: 1= 1's complement for CRC_SUM 0=No 1's complement for CRC_SUM	0
31:6	Reserved	Always 0 when read	0x0000000

19.6.2 CRC seed register

Table 209. CRC seed register (SEED, address 0x5000 0004) bit description

Bit	Symbol	Description	Reset value
31:0	CRC_SEED	A write access to this register will load CRC seed value to CRC_SUM register with selected bit order and 1's complement pre-processes. **Remark:** A write access to this register will overrule the CRC calculation in progresses.	0x0000 FFFF

19.6.3 CRC checksum register

This register is a Read-only register containing the most recent checksum. The read request to this register is automatically delayed by a finite number of wait states until the results are valid and the checksum computation is complete.

Table 210. CRC checksum register (SUM, address 0x5000 0008) bit description

Bit	Symbol	Description	Reset value
31:0	CRC_SUM	The most recent CRC sum can be read through this register with selected bit order and 1's complement post-processes.	0x0000 FFFF

19.6.4 CRC data register

This register is a Write-only register containing the data block for which the CRC sum will be calculated.

Table 211. CRC data register (WR_DATA, address 0x5000 0008) bit description

Bit	Symbol	Description	Reset value
31:0	CRC_WR_DATA	Data written to this register will be taken to perform CRC calculation with selected bit order and 1's complement pre-process. Any write size 8, 16 or 32-bit are allowed and accept back-to-back transactions.	-

19.7 Functional description

The following sections describe the register settings for each supported CRC standard:

19.7.1 CRC-CCITT set-up

Polynomial = $x^{16} + x^{12} + x^5 + 1$

Seed Value = 0xFFFF

Bit order reverse for data input: NO

1's complement for data input: NO

Bit order reverse for CRC sum: NO

1's complement for CRC sum: NO

CRC_MODE = 0x0000 0000

CRC_SEED = 0x0000 FFFF

19.7.2 CRC-16 set-up

Polynomial = $x^{16} + x^{15} + x^2 + 1$

Seed Value = 0x0000

Bit order reverse for data input: YES

1's complement for data input: NO

Bit order reverse for CRC sum: YES

1's complement for CRC sum: NO

CRC_MODE = 0x0000 0015

CRC_SEED = 0x0000 0000

19.7.3 CRC-32 set-up

Polynomial = $x^{32}+ x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Seed Value = 0xFFFF FFFF

Bit order reverse for data input: YES

1's complement for data input: NO

Bit order reverse for CRC sum: YES

1's complement for CRC sum: YES

CRC_MODE = 0x0000 0036

CRC_SEED = 0xFFFF FFFF

20.1 How to read this chapter

The flash controller is identical on all LPC800 parts.

20.2 Features

- Controls flash access time.
- Provides registers for flash signature generation.

20.3 General description

The flash controller is accessible for programming flash wait states and for generating the the flash signature.

20.4 Register description

Table 212. Register overview: FMC (base address 0x4004 0000)

Name	Access	Address offset	Description	Reset value	Reference
FLASHCFG	R/W	0x010	Flash configuration register	<tbd>	Table 213
FMSSTART	R/W	0x020	Signature start address register	0	Table 214
FMSSTOP	R/W	0x024	Signature stop-address register	0	Table 215
FMSW0	R	0x02C	Signature word	-	Table 216

20.4.1 Flash configuration register

Access to the flash memory can be configured independently of the system frequency by writing to the FLASHCFG register.

Remark: When using the Power API, do not change the waitstates in efficiency, low-current, or performance modes.

Table 213. Flash configuration register (FLASHCFG, address 0x4004 0010) bit description

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

20.4.2 Flash signature start address register

Table 214. Flash Module Signature Start register (FMSSTART - 0x4004 0020) bit description

Bit	Symbol	Description	Reset value
16:0	START	Signature generation start address (corresponds to AHB byte address bits[20:4]).	0
31:17	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

20.4.3 Flash signature stop address register

Table 215. Flash Module Signature Stop register (FMSSTOP - 0x4004 0024) bit description

Bit	Symbol	Value	Description	Reset value
16:0	STOPA		Stop address for signature generation (the word specified by STOPA is included in the address range). The address is in units of memory words, not bytes.	0
30:17	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
31	STRTBIST		When this bit is written to 1, signature generation starts. At the end of signature generation, this bit is automatically cleared.	0

20.4.4 Flash signature generation result register

The signature generation result register returns the flash signature produced by the embedded signature generator.

The generated flash signature can be used to verify the flash memory contents. The generated signature can be compared with an expected signature and thus makes saves time and code space. The method for generating the signature is described in Section 20.5.1.

Table 216. FMSW0 register bit description (FMSW0, address: 0x4004 002C)

Bit	Symbol	Description	Reset value
31:0	SIG	32-bit signature.	-

20.5 Functional description

20.5.1 Flash signature generation

The flash module contains a built-in signature generator. This generator can produce a 32-bit signature from a range of flash memory. A typical usage is to verify the flashed contents against a calculated signature (e.g. during programming).

The address range for generating a signature must be aligned on flash-word boundaries, i.e. 32-bit boundaries. Once started, signature generation completes independently. While signature generation is in progress, the flash memory cannot be accessed for other purposes, and an attempted read will cause a wait state to be asserted until signature

generation is complete. Code outside of the flash (e.g. internal RAM) can be executed during signature generation. This can include interrupt services, if the interrupt vector table is re-mapped to memory other than the flash memory. The code that initiates signature generation should also be placed outside of the flash memory.

20.5.1.1 Signature generation address and control registers

These registers control automatic signature generation. A signature can be generated for any part of the flash memory contents. The address range to be used for generation is defined by writing the start address to the signature start address register (FMSSTART) and the stop address to the signature stop address register (FMSSTOP. The start and stop addresses must be aligned to 32-bit boundaries.

Signature generation is started by setting the STRTBIST bit in the FMSSTOP register. Setting the STRTBIST bit is typically combined with the signature stop address in a single write.

Table 214 and Table 215 show the bit assignments in the FMSSTART and FMSSTOP registers respectively.

20.5.1.2 Signature generation

A signature can be generated for any part of the flash contents. The address range to be used for signature generation is defined by writing the start address to the FMSSTART register, and the stop address to the FMSSTOP register.

The signature generation is started by writing a 1 to the SIG_START bit in the FMSSTOP register. Starting the signature generation is typically combined with defining the stop address, which is done in the STOP bits of the same register.

The time that the signature generation takes is proportional to the address range for which the signature is generated. Reading of the flash memory for signature generation uses a self-timed read mechanism and does not depend on any configurable timing settings for the flash. A safe estimation for the duration of the signature generation is:

Duration = int((60 / tcy) + 3) x (FMSSTOP - FMSSTART + 1)

When signature generation is triggered via software, the duration is in AHB clock cycles, and tcy is the time in ns for one AHB clock. The SIG_DONE bit in FMSTAT can be polled by software to determine when signature generation is complete.

After signature generation, a 32-bit signature can be read from the FMSW0 register. The 32-bit signature reflects the corrected data read from the flash and the flash parity bits and check bit values.

20.5.1.3 Content verification

The signature as it is read from the FMSW0 register must be equal to the reference signature. The following pseudo-code shows the algorithm to derive the reference signature:

```
sign = 0
FOR address = FMSSTART.START to FMSSTOP.STOPA
{
        FOR i = 0 TO 30
        {
```

```
                nextSign[i] = f_Q[address][i] XOR sign[i + 1]
            }
            nextSign[31] = f_Q[address][31] XOR sign[0] XOR sign[10] XOR sign[30] XOR sign[31]
            sign = nextSign
    }
    signature32 = sign
```

21.1 How to read this chapter

The boot loader is identical for all parts. The Boot ROM implementation changes with the chip version. See Section 21.3.1.

21.2 Features

- 8 kB on-chip boot ROM
- Contains the boot loader with In-System Programming (ISP) facility and the following APIs:
  - In Application Programming (IAP) of flash memory
  - Power profiles for optimizing power consumption and system performance
  - USART drivers
  - I2C drivers

21.3 Basic configuration

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

21.3.1 Boot loader versions

The LPC800 boot loader is updated with a new chip version. You can determine the boot loader version using the ISP command Read Boot code version (see Section 22.5.1.12) or from the part marking.

Table 217. Boot loader versions

Boot loader version	Marking	API	Description
v13.1 (initial version)	1A	ISP/IAP	The following deviations from the specification apply: • The the IAP erase page command allows only single-page erase. The start page parameter must the same as the end page parameter. See Table 252. • Code SECTOR_NOT_PREPARED_FORWRITE_OPERATION in ISP command C (Write RAM to flash) is not returned. See Table 231. • The ISP mode uses the USART0 interface for communication. If USART0 is used in an application, reset USART0 (see Table 7) before using the IAP command 57 (Reinvoke ISP). See Table 250.
		UART	The following deviations from the specification apply: • UART synchronous mode not supported. • API functions uart_put_line and uart_get_line do not return an interrupt on error. See Table 285 and Table 286. • UART API return codes are numbered 0x0007 0001 to 0x0007 0005.
		I2C	No changes.
		Power profiles	No changes.

Table 217. Boot loader versions

Boot loader version	Marking	API	Description
v13.2	2A	ISP/IAP	The following updates compared to v13.1 apply: • The IAP erase page command allows multiple-page erase. Any start page number that is smaller or equal to the end page number is allowed as start page in the IAP erase page command. See Table 252. • Code SECTOR_NOT_PREPARED_FOR WRITE_OPERATION in ISP command C (Write RAM to flash) is returned. See Table 231. • IAP command 57 (Reinvoke ISP) can be called without resetting USART0 first. • ISP command S (Read CRC checksum) added. See Table 240.
		UART	The following updates compared to v13.1 apply: • UART synchronous mode supported. • API functions uart_put_line and uart_get_line do return an interrupt on error. See Table 285 and Table 286. • UART API return codes are numbered 0x0008 0001 to 0x0008 0005. See Table 288.
		I2C	No changes.
		Power profiles	No changes.

21.4 Pin description

When pin PIO0_1 is pulled LOW on reset, the part enters ISP mode and the ISP command handler starts up. In ISP mode, pins PIO0_0 is connected to function U0_RXD and pin PIO0_4 is connected to function U0_TXD on the USART0 block.

21.5 General description

21.5.1 Boot loader

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

The boot loader code is executed every time the part is powered on or reset. The boot loader can execute the ISP command handler or the user application code. A LOW level after reset at the PIO0_1 pin is considered as an external hardware request to start the ISP command handler via USART.

For details on the boot process, see Section 21.6.2 “Boot process”.

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

Assuming that power supply pins are on their nominal levels when the rising edge on $\overline{\text{RESET}}$ pin is generated, it may take up to <tbd>3 ms before PIO0_1 is sampled and the decision whether to continue with user code or ISP handler 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 22.4.3 “Code Read Protection (CRP)”).

21.5.2 ROM-based APIs

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

The structure of the boot ROM APIs is shown in Figure 41.

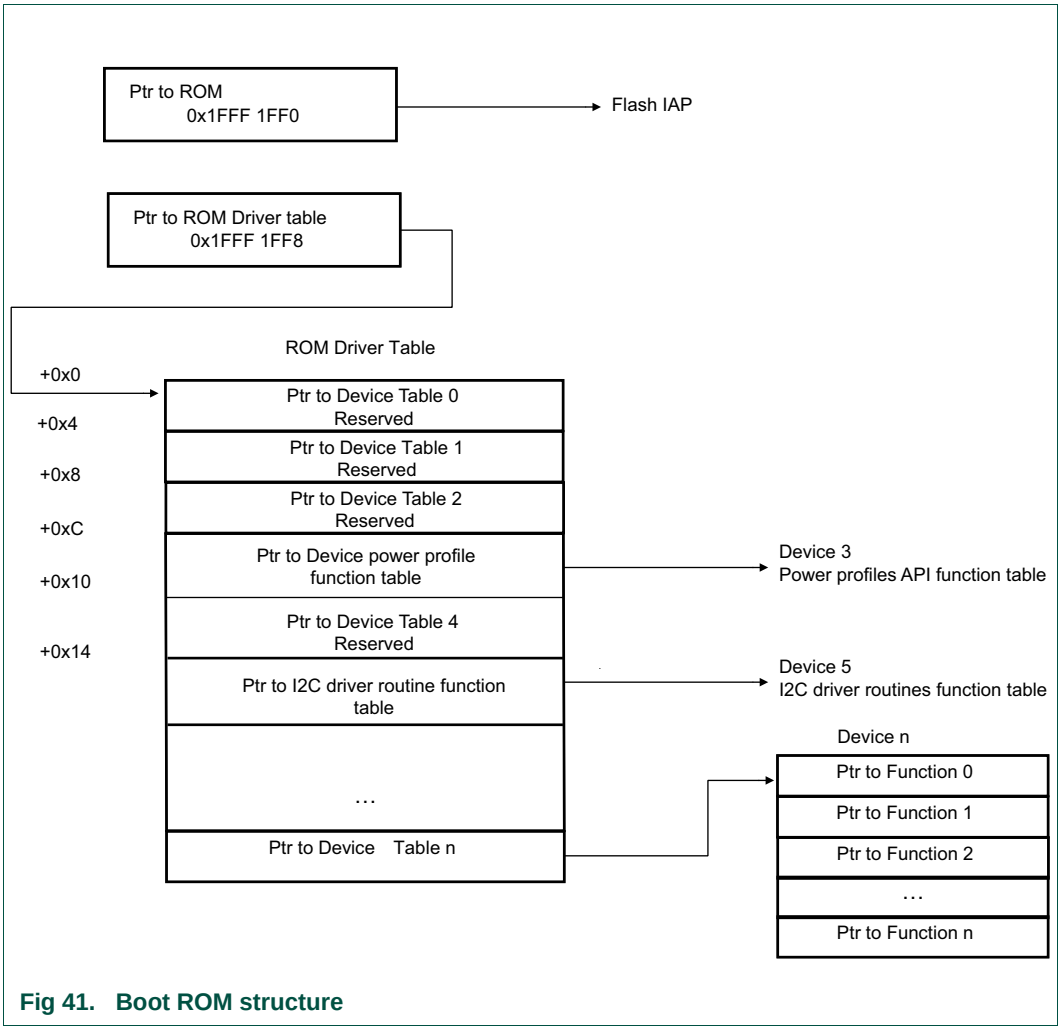


Fig 41. Boot ROM structure

Table 218. API calls

API	Description	Reference
Flash IAP	Flash In-Application programming	Table 242
Power profiles API	Configure system clock and power consumption	Table 255
I2C driver	I2C ROM Driver	Table 258
UART driver	USART ROM Driver	Table 279

21.6 Functional description

21.6.1 Memory map after any reset

The boot block is 8 kB in size. The boot block is located in the memory region starting from the address 0x1FFF 0000. The 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.

21.6.2 Boot process

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

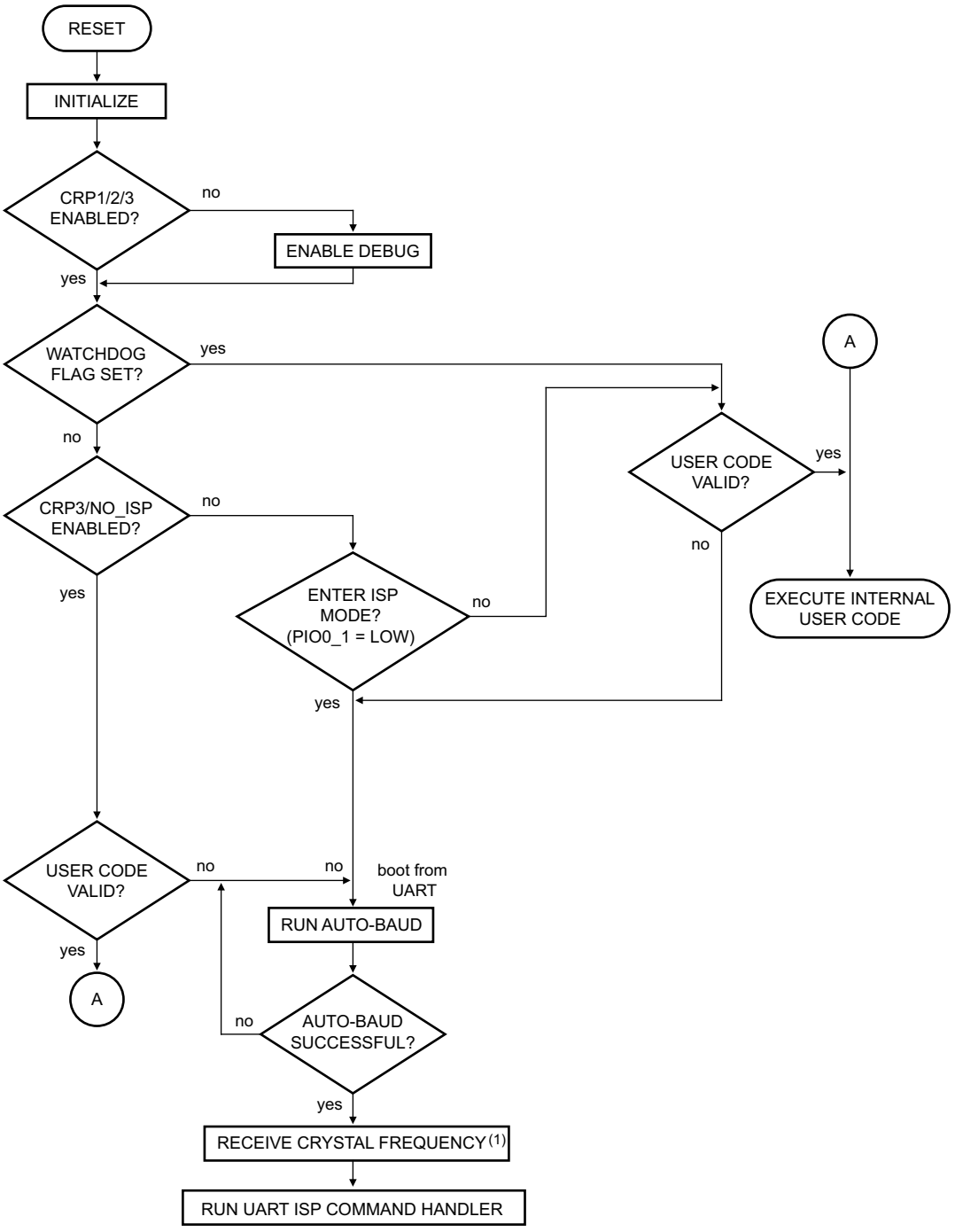
The reserved Cortex-M0+ exception vector location 7 (offset 0x0000 001C in the vector table) should contain the 2’s complement of the check-sum of table entries 0 through 6. This causes the checksum of the first 8 table entries to be 0. The bootloader code checksums the first 8 locations in sector 0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

If the signature is not valid, the auto-baud routine synchronizes with the host via serial port USART0. The host should send a ’?’ (0x3F) as a synchronization character and wait for a response. The host side serial port settings should be 8 data bits, 1 stop bit and no parity. The auto-baud routine measures the bit time of the received synchronization character in terms of its own frequency (the 12 MHz IRC frequency) and programs the baud rate generator of the serial port. It also sends an ASCII string ("Synchronized<CR><LF>") to the host. In response, the host should send the same string ("Synchronized<CR><LF>").

The boot loader auto-baud routine looks at the received characters to verify synchronization. If synchronization is verified then "OK<CR><LF>" string is sent to the host. The host should respond by sending the crystal frequency (in kHz) at which the part is running. The response is required for backward compatibility of the boot loader code and, on the LPC800, is ignored. The boot loader configures the part to run at the 12 MHz IRC frequency.

Once the crystal frequency response is received, the part is initialized and the ISP command handler is invoked. For safety reasons an "Unlock" command is required before executing the commands resulting in flash erase/write operations and the "Go" command. The rest of the commands can be executed without the unlock command. The Unlock command is required to be executed once per ISP session. The Unlock command is explained in Table 225 “UART ISP Unlock command”.

21.6.3 Boot process flowchart



(1) This step is included for backward compatibility and the response is ignored by the boot loader.

Fig 42. Boot process flowchart

22.1 How to read this chapter

See Table 219 for different flash configurations.

Table 219. LPC800 flash configurations

Type number	Flash
LPC810M021FN8	4 kB
LPC811M001FDH16	8 kB
LPC812M101FDH16	16 kB
LPC812M101FD20	16 kB
LPC812M101FDH20	16 kB

22.2 Features

- In-System Programming: In-System programming (ISP) is programming or reprogramming the on-chip flash memory, using the bootloader software and UART serial port.
- In-Application Programming: In-Application (IAP) programming is performing erase and write operation on the on-chip flash memory, as directed by the end-user application code.
- You can use ISP and IAP when the part resides in the end-user board.
- Flash page write and erase supported.

22.3 Pin description

When pin PIO0_1 is pulled LOW on reset, the part enters ISP mode and the ISP command handler starts up. In ISP mode, pin PIO0_0 is connected to function U0_RXD and pin PIO0_4 is connected to function U0_TXD on the USART0 block.

22.4 General description

22.4.1 Flash configuration

Most IAP and ISP commands operate on sectors and specify sector numbers. In addition a page erase command is supported. The following table shows the correspondence between page numbers, sector numbers, and memory addresses.

The size of a sector is 1 kB and the size of a page is 64 Byte. One sector contains 16 pages.

Table 220. LPC800 flash configuration

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

22.4.2 Flash content protection mechanism

The part is equipped with the Error Correction Code (ECC) capable Flash memory. The purpose of an error correction module is twofold:

The ECC first decodes data words read from the memory into output data words. Then, the ECC encodes data words to be written to the memory. The error correction capability consists of single bit error correction with Hamming code.

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

Whenever the CPU requests a read from the user accessible Flash, both 32 bits of raw data containing the specified memory location and the matching ECC byte are evaluated. If the ECC mechanism detects a single error in the fetched data, a correction will be applied before data are provided to the CPU. When a write request into the user accessible Flash is made, writing the user specified content is accompanied by a matching ECC value calculated and stored in the ECC memory.

When a sector of Flash memory is erased, the corresponding ECC bits are also erased. Once a 6-bit ECC is written, it can not be updated unless it is erased first. Therefore, for the implemented ECC mechanism to perform properly, data must be written into the flash memory in groups of 4 bytes (or multiples of 4), aligned as described above.

22.4.3 Code Read Protection (CRP)

Code Read Protection is a mechanism that allows the user to enable different levels of security in the system so that access to the on-chip flash and use of the ISP can be restricted. When needed, CRP is invoked by programming a specific pattern in flash location at 0x0000 02FC. IAP commands are not affected by the code read protection.

Important: any CRP change becomes effective only after the device has gone through a power cycle.

Table 221. Code Read Protection 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 should not access RAM below 0x1000 0300. Access to addresses below 0x1000 0200 is disabled. • Copy RAM to flash command can not write to Sector 0. • Erase command can erase Sector 0 only when all sectors are selected for erase. • Compare command is disabled. • Read Memory command is disabled. This mode is useful when CRP is required and flash field updates are needed but all sectors can not be erased. Since compare command is disabled in case of partial updates the secondary loader should implement checksum mechanism to verify the integrity of the flash.
CRP2	0x87654321	Access to chip via the SWD pins is disabled. The following ISP commands are disabled: • Read Memory • Write to RAM • Go • Copy RAM to flash • Compare When CRP2 is enabled the ISP erase command only allows erasure of all user sectors.
CRP3	0x43218765	Access to chip via the SWD pins is disabled. ISP entry by pulling 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 UART. **Caution: If CRP3 is selected, no future factory testing can be performed on the device.**

Table 222. Code Read Protection hardware/software interaction

CRP option	User Code Valid	PIO0_1 pin at reset	SWD enabled	Part enters ISP mode	partial flash update in ISP mode
None	No	x	Yes	Yes	Yes
None	Yes	High	Yes	No	NA
None	Yes	Low	Yes	Yes	Yes
CRP1	Yes	High	No	No	NA
CRP1	Yes	Low	No	Yes	Yes

Table 222. Code Read Protection hardware/software interaction

CRP option	User Code Valid	PIO0_1 pin at reset	SWD enabled	Part 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 223. 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.

22.4.3.1 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.

22.5 API description

22.5.1 UART ISP commands

The following commands are accepted by the ISP command handler. Detailed status codes are supported for each command. The command handler sends the return code INVALID_COMMAND when an undefined command is received. Commands and return codes are in ASCII format.

CMD_SUCCESS is sent by ISP command handler only when received ISP command has been completely executed and the new ISP command can be given by the host. Exceptions from this rule are "Set Baud Rate", "Write to RAM", "Read Memory", and "Go" commands.

Table 224. UART ISP command summary

ISP Command	Usage	Described in
Unlock	U <Unlock Code>	Table 225
Set Baud Rate	B <Baud Rate> <stop bit>	Table 226
Echo	A <setting>	Table 227
Write to RAM	W <start address> <number of bytes>	Table 228
Read Memory	R <address> <number of bytes>	Table 229
Prepare sector(s) for write operation	P <start sector number> <end sector number>	Table 230
Copy RAM to flash	C <Flash address> <RAM address> <number of bytes>	Table 231
Go	G <address> <Mode>	Table 232
Erase sector(s)	E <start sector number> <end sector number>	Table 233
Blank check sector(s)	I <start sector number> <end sector number>	Table 234
Read Part ID	J	Table 235
Read Boot code version	K	Table 237
Compare	M <address1> <address2> <number of bytes>	Table 238
ReadUID	N	Table 239
Read CRC checksum	S <address> <number of bytes>	Table 240

22.5.1.1 Unlock <Unlock code>

Table 225. UART 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.

22.5.1.2 Set Baud Rate <Baud Rate> <stop bit>

Table 226. UART ISP Set Baud Rate command

Command	B
Input	Baud Rate: 9600 | 19200 | 38400 | 57600 | 115200 Stop bit: 1 | 2
Return Code	CMD_SUCCESS | INVALID_BAUD_RATE | INVALID_STOP_BIT | PARAM_ERROR
Description	This command is used to change the baud rate. The new baud rate is effective after the command handler sends the CMD_SUCCESS return code.
Example	"B 57600 1<CR><LF>" sets the serial port to baud rate 57600 bps and 1 stop bit.

22.5.1.3 Echo <setting>

Table 227. UART ISP Echo command

Command	A
Input	Setting: ON = 1 | OFF = 0
Return Code	CMD_SUCCESS | PARAM_ERROR
Description	The default setting for echo command is ON. When ON the ISP command handler sends the received serial data back to the host.
Example	"A 0<CR><LF>" turns echo off.

22.5.1.4 Write to RAM <start address> <number of bytes>

The host should send the plain binary code after receiving the CMD_SUCCESS return code. This ISP command handler responds with “OK<CR><LF>” when the transfer has finished.

Table 228. UART ISP Write to RAM command

Command	W
Input	**Start Address:** RAM address where data bytes are to be written. This address should be a word boundary. **Number of Bytes:** Number of bytes to be written. Count should be a multiple of 4
Return Code	CMD_SUCCESS | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to download data to RAM. This command is blocked when code read protection levels 2 or 3 are enabled. Writing to addresses below 0x1000 0300 is disabled for CRP1.
Example	"W 268436224 4<CR><LF>" writes 4 bytes of data to address 0x1000 0300.

22.5.1.5 Read Memory <address> <number of bytes>

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

Table 229. UART ISP Read Memory command

Command	R
Input	**Start Address:** Address from where data bytes are to be read. This address should be a word boundary. **Number of Bytes:** Number of bytes to be read. Count should be a multiple of 4.
Return Code	CMD_SUCCESS followed by <actual data (plain binary)> | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not a multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to read data from RAM or flash memory. This command is blocked when code read protection is enabled.
Example	"R 268435456 4<CR><LF>" reads 4 bytes of data from address 0x1000 0000.

22.5.1.6 Prepare sector(s) for write operation <start sector number> <end sector number>

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

Table 230. UART ISP Prepare sector(s) for write operation command

Command	P
Input	**Start Sector Number** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | PARAM_ERROR
Description	This command must be executed before executing "Copy RAM to flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. The boot block can not be prepared by this command. To prepare a single sector use the same "Start" and "End" sector numbers.
Example	"P 0 0<CR><LF>" prepares the flash sector 0.

22.5.1.7 Copy RAM to flash <Flash address> <RAM address> <no of bytes>

When writing to the flash, the following limitations apply:

1. The smallest amount of data that can be written to flash by the copy RAM to flash command is 64 byte (equal to one page).
2. One page consists of 16 flash words (lines), and the smallest amount that can be modified per flash write is one flash word (one line). This limitation follows from the application of ECC to the flash write operation, see Section 22.4.2.

3. To avoid write disturbance (a mechanism intrinsic to flash memories), an erase should be performed after following 16 consecutive writes inside the same page. Note that the erase operation then erases the entire sector.

   Remark: Once a page has been written to 16 times, it is still possible to write to other pages within the same sector without performing a sector erase (assuming that those pages have been erased previously).

Table 231. UART ISP Copy RAM to flash command

Command	C
Input	**Flash Address (DST):** Destination flash address where data bytes are to be written. The destination address should be a 64 byte boundary. **RAM Address (SRC):** Source RAM address from where data bytes are to be read. **Number of Bytes:** Number of bytes to be written. Should be 64 | 128 | 256 | 512 | 1024.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not on word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 64 | 128 | 256 | 512 | 1024) | SECTOR_NOT_PREPARED_FOR WRITE_OPERATION | BUSY | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to program the flash memory. The "Prepare Sector(s) for Write Operation" command should precede this command. The affected sectors are automatically protected again once the copy command is successfully executed. The boot block cannot be written by this command. This command is blocked when code read protection is enabled.
Example	"C 0 268437504 512<CR><LF>" copies 512 bytes from the RAM address 0x1000 0800 to the flash address 0.

22.5.1.8 Go <address> <mode>

Table 232. UART ISP Go command

Command	G
Input	**Address:** Flash or RAM address from which the code execution is to be started. This address should be on a word boundary. **Mode:** T (Execute program in Thumb Mode).
Return Code	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to execute a program residing in RAM or flash memory. It may not be possible to return to the ISP command handler once this command is successfully executed. This command is blocked when code read protection is enabled. The command must be used with an address of 0x0000 0200 or greater.
Example	"G 512 T<CR><LF>" branches to address 0x0000 0200 in Thumb mode.

22.5.1.9 Erase sector(s) <start sector number> <end sector number>

Table 233. UART ISP Erase sector command

Command	E
Input	**Start Sector Number** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to erase one or more sector(s) of on-chip flash memory. The boot block can not be erased using this command. This command only allows erasure of all user sectors when the code read protection is enabled.
Example	"E 2 3<CR><LF>" erases the flash sectors 2 and 3.

22.5.1.10 Blank check sector(s) <sector number> <end sector number>

Table 234. UART ISP Blank check sector command

Command	I
Input	**Start Sector Number:** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | SECTOR_NOT_BLANK (followed by <Offset of the first non blank word location> <Contents of non blank word location>) | INVALID_SECTOR | PARAM_ERROR
Description	This command is used to blank check one or more sectors of on-chip flash memory. **Blank check on sector 0 always fails as first 64 bytes are re-mapped to flash boot block.** When CRP is enabled, the blank check command returns 0 for the offset and value of sectors which are not blank. Blank sectors are correctly reported irrespective of the CRP setting.
Example	"I 2 3<CR><LF>" blank checks the flash sectors 2 and 3.

22.5.1.11 Read Part Identification number

Table 235. UART ISP Read Part Identification command

Command	J
Input	None.
Return Code	CMD_SUCCESS followed by part identification number in ASCII (see Table 236).
Description	This command is used to read the part identification number.

Table 236. Part identification numbers

Device	Hex coding
LPC810M021FN8	0x0000 8100
LPC811M001FDH16	0x0000 8110
LPC812M101FDH16	0x0000 8120
LPC812M101FD20	0x0000 8121
LPC812M101FDH20	0x0000 8122

22.5.1.12 Read Boot code version number

Table 237. UART ISP Read Boot Code version number command

Command	K
Input	None
Return Code	CMD_SUCCESS followed by 2 bytes of boot code version number in ASCII format. It is to be interpreted as <byte1(Major)>.<byte0(Minor)>.
Description	This command is used to read the boot code version number.

22.5.1.13 Compare <address1> <address2> <no of bytes>

Table 238. UART ISP Compare command

Command	M
Input	**Address1 (DST):** Starting flash or RAM address of data bytes to be compared. This address should be a word boundary. **Address2 (SRC):** Starting flash or RAM address of data bytes to be compared. This address should be a word boundary. **Number of Bytes:** Number of bytes to be compared; should be a multiple of 4.
Return Code	CMD_SUCCESS | (Source and destination data are equal) COMPARE_ERROR | (Followed by the offset of first mismatch) COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED | PARAM_ERROR
Description	This command is used to compare the memory contents at two locations.
Example	"M 8192 268468224 4<CR><LF>" compares 4 bytes from the RAM address 0x1000 8000 to the 4 bytes from the flash address 0x2000.

22.5.1.14 ReadUID

Table 239. UART ISP ReadUID command

Command	N
Input	None
Return Code	CMD_SUCCESS followed by four 32-bit words of E-sort test information in ASCII format. The word sent at the lowest address is sent first.
Description	This command is used to read the unique ID.

22.5.1.15 Read CRC checksum <address> <no of bytes>

Get the CRC checksum of a block of RAM or flash. CMD_SUCCESS followed by 8 bytes of CRC checksum in ASCII format.

The checksum is calculated as follows:

CRC-32 polynomial: x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

Seed Value: 0xFFFF FFFF

No bit order reverse for data input

No 1’s complement for data input

No bit order reverse for CRC sum

No 1’s complement for CRC sum

Table 240. UART ISP Read CRC checksum command

Command	S
Input	**Address:** The data are read from this address for CRC checksum calculation. This address must be on a word boundary. **Number of Bytes:** Number of bytes to be calculated for the CRC checksum; must be a multiple of 4.
Return Code	CMD_SUCCESS followed by data in plain binary format ADDR_ERROR (address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (byte count is not a multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to read the CRC checksum of a block of RAM or flash memory. This command is blocked when code read protection is enabled.
Example	"S 268436736 4<CR><LF>" reads the CRC checksum for 4 bytes of data from address 0x1000 0500. If checksum value is 0xCBF43926, then the host will receive: "3421780262 <CR><LF>"

22.5.1.16 UART ISP Return Codes

Table 241. UART ISP Return Codes Summary

Return Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable.
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid or end sector number is greater than start sector number.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data not equal.
11	BUSY	Flash programming hardware interface is busy.
12	PARAM_ERROR	Insufficient number of parameters or invalid parameter.
13	ADDR_ERROR	Address is not on word boundary.

Table 241. UART ISP Return Codes Summary

Return Code	Mnemonic	Description
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.

22.5.2 IAP commands

For in application programming the IAP routine should be called with a word pointer in register r0 pointing to memory (RAM) containing command code and parameters. Result of the IAP command is returned in the result table pointed to by register r1. The user can reuse the command table for result by passing the same pointer in registers r0 and r1. The parameter table should be big enough to hold all the results in case the number of results are more than number of parameters. Parameter passing is illustrated in the Figure 43. 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 242. IAP Command Summary

IAP Command	Command Code	Described in
Prepare sector(s) for write operation	50 (decimal)	Table 243
Copy RAM to flash	51 (decimal)	Table 244
Erase sector(s)	52 (decimal)	Table 245
Blank check sector(s)	53 (decimal)	Table 246
Read Part ID	54 (decimal)	Table 247
Read Boot code version	55 (decimal)	Table 248
Compare	56 (decimal)	Table 249
Reinvoke ISP	57 (decimal)	Table 250
Read UID	58 (decimal)	Table 251
Erase page	59 (decimal)	Table 252

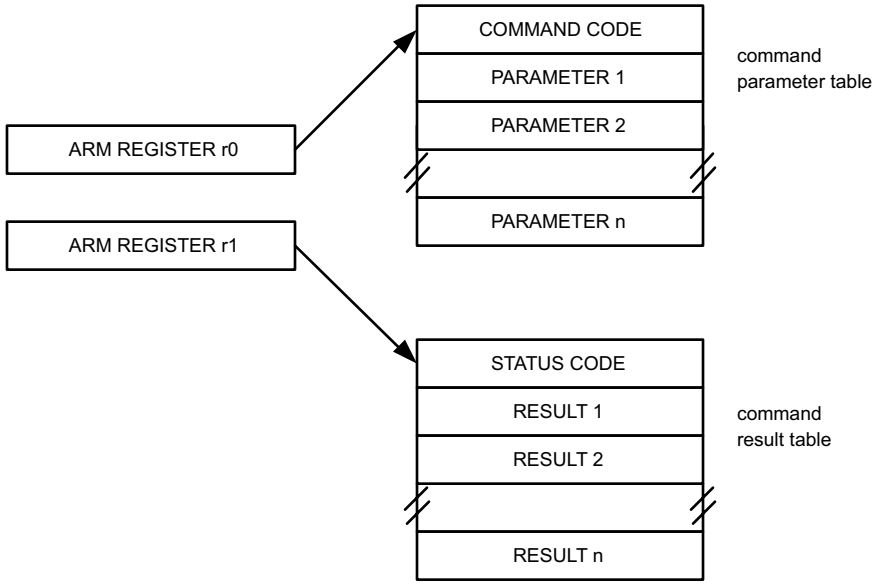


Fig 43. IAP parameter passing

22.5.2.1 Prepare sector(s) for write operation (IAP)

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

Table 243. 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.

22.5.2.2 Copy RAM to flash (IAP)

See Section 22.5.1.4 for limitations on the write-to-flash process.

Table 244. IAP Copy RAM to flash command

Command	Copy RAM to flash
Input	**Command code: 51 (decimal)** **Param0(DST):** Destination flash address where data bytes are to be written. This address should be a 64 byte boundary. **Param1(SRC):** Source RAM address from which data bytes are to be read. This address should be a word boundary. **Param2:** Number of bytes to be written. Should be 64 | 128 | 256 | 512 | 1024. **Param3:** 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.

22.5.2.3 Erase Sector(s) (IAP)

Table 245. 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.

22.5.2.4 Blank check sector(s) (IAP)

Table 246. 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.

22.5.2.5 Read Part Identification number (IAP)

Table 247. IAP Read Part Identification command

Command	Read part identification number
Input	**Command code: 54 (decimal)** **Parameters:** None
Return Code	CMD_SUCCESS
Result	**Result0:** Part Identification Number.
Description	This command is used to read the part identification number.

22.5.2.6 Read Boot code version number (IAP)

Table 248. 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. Read as <byte1(Major)>.<byte0(Minor)>
Description	This command is used to read the boot code version number.

22.5.2.7 Compare <address1> <address2> <no of bytes> (IAP)

Table 249. 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.

22.5.2.8 Reinvoke ISP (IAP)

Table 250. IAP 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, and configures USART0 pins U0_RXD and U0_TXD. This command may be used when a valid user program is present in the internal flash memory and the PIO0_1 pin is not accessible to force the ISP mode.

22.5.2.9 ReadUID (IAP)

Table 251. 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.

22.5.2.10 Erase page

Table 252. IAP Erase page command

Command	Erase page
Input	**Command code: 59 (decimal)** **Param0:** Start page number. **Param1:** End page number (should be greater than or equal to start page) **Param2:** System Clock Frequency (CCLK) in kHz.
Return Code	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
Result	None
Description	This command is used to erase a page or multiple pages of on-chip flash memory. To erase a single page use the same "start" and "end" page numbers.

22.5.2.11 IAP Status Codes

Table 253. 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.

Table 253. IAP Status Codes Summary

Status Code	Mnemonic	Description
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.

22.6 Functional description

22.6.1 UART Communication protocol

All UART ISP commands should be sent as single ASCII strings. Strings should be terminated with Carriage Return (CR) and/or Line Feed (LF) control characters. Extra <CR> and <LF> characters are ignored. All ISP responses are sent as <CR><LF> terminated ASCII strings. Data is sent and received in plain binary format.

22.6.1.1 UART ISP command format

"Command Parameter_0 Parameter_1 ... Parameter_n<CR><LF>" "Data" (Data only for Write commands).

22.6.1.2 UART ISP response format

"Return_Code<CR><LF>Response_0<CR><LF>Response_1<CR><LF> ... Response_n<CR><LF>" "Data" (Data only for Read commands).

22.6.1.3 UART ISP data format

The data stream is in plain binary format.

22.6.2 Memory and interrupt use for ISP and IAP

22.6.2.1 Interrupts during UART ISP

The boot block interrupt vectors located in the boot block of the flash are active after any reset.

22.6.2.2 Interrupts during IAP

The on-chip flash memory is not accessible during erase/write operations. When the user application code starts executing the interrupt vectors from the user flash area are active. Before making any IAP call, either disable the interrupts or ensure that the user interrupt vectors are active in RAM and that the interrupt handlers reside in RAM. The IAP code does not use or disable interrupts.

22.6.2.3 RAM used by ISP command handler

The stack of ISP commands is located at 0x1000 0270. The maximum stack usage is 540 byte and grows downwards.

22.6.2.4 RAM used by IAP command handler

The maximum stack usage in the user allocated stack space is 148 bytes and grows downwards.

22.6.3 Debugging

22.6.3.1 Comparing flash images

Depending on the debugger used and the IDE debug settings, the memory that is visible when the debugger connects might be the boot ROM, the internal SRAM, or the flash. To help determine which memory is present in the current debug environment, check the value contained at flash address 0x0000 0004. This address contains the entry point to the code in the ARM Cortex-M0+ vector table, which is the bottom of the boot ROM, the internal SRAM, or the flash memory respectively.

Table 254. 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

22.6.3.2 Serial Wire Debug (SWD) flash programming interface

Debug tools can write parts of the flash image to RAM and then execute the IAP call "Copy RAM to flash" repeatedly with proper offset.

23.1 How to read this chapter

The power profiles are available for all LPC800 parts.

23.2 Features

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

23.3 General description

The power consumption in Active and Sleep modes can be optimized for the application through simple calls to the power profile. The power configuration routine configures the LPC800 for one of the following power modes:

- Default mode corresponding to power configuration after reset.
- CPU performance mode corresponding to optimized processing capability.
- Efficiency mode corresponding to optimized balance of current consumption and CPU performance.
- Low-current mode corresponding to lowest power consumption.

In addition, the power profile includes routines to select the optimal PLL settings for a given system clock and PLL input clock.

Remark: Disable all interrupts before making calls to the power profile API. You can re-enable the interrupts after the power profile API calls have completed.

The API calls to the ROM are performed by executing functions which are pointed by a pointer within the ROM Driver Table. Figure 44 shows the pointer structure used to call the Power Profiles API.

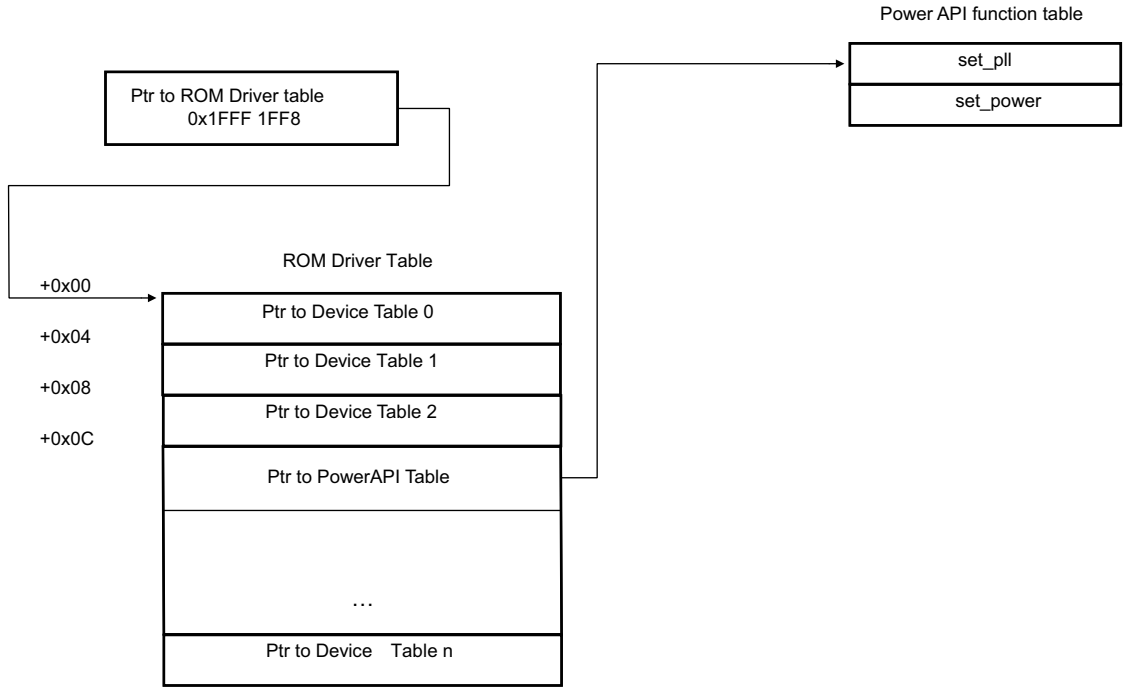


Fig 44. Power profiles pointer structure

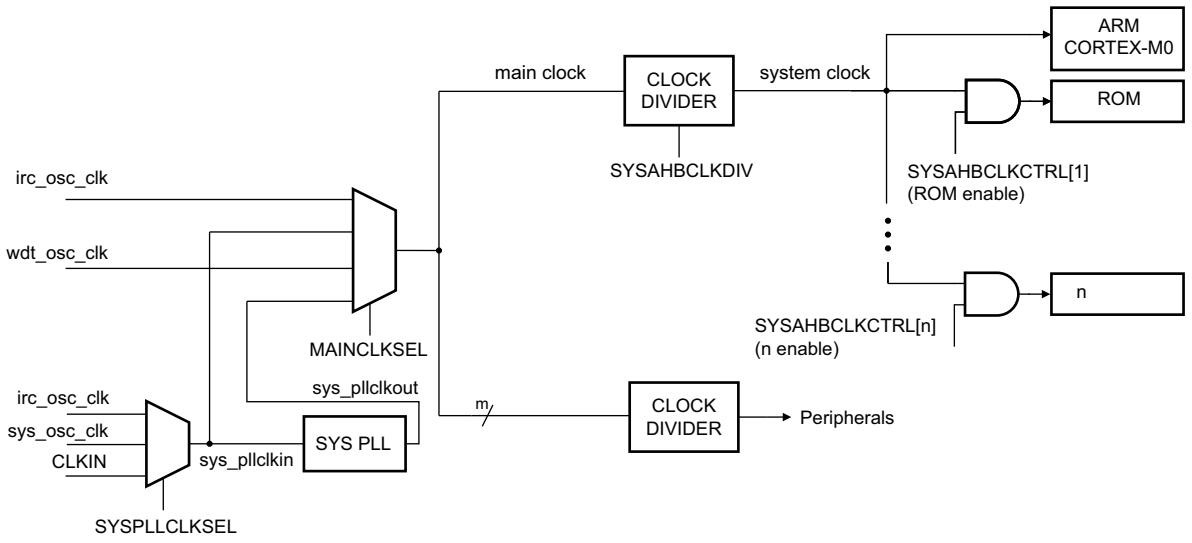


Fig 45. LPC800 clock configuration for power API use

23.4 API description

The power profile API provides functions to configure the system clock and optimize the system setting for lowest power consumption.

Table 255. Power profile API calls

API call	Description	Reference
set_pll(command, result)	Power API set pll routine	Table 256
set_power(command, result)	Power API set power routine	Table 257

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];
```

23.4.1 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 13), the main clock source must be set to the input clock to the system PLL (Table 15). and the system/AHB clock divider must be set to 1 (Table 17).

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 256. 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 45. For more details see Figure 3.

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

set_pll configures a setup in which the main clock does not exceed 30 MHz (see Figure 45). It easily finds a solution when the ratio between the expected system clock and the system PLL input frequency is an integer value, but it can also find solutions in other cases.

The system PLL input frequency (*Param0*) must be between 10000 to 25000 kHz (10 MHz to 25 MHz) inclusive. The expected system clock (*Param1*) must be between 1 and 30000 kHz inclusive. If either of these requirements is not met, *set_pll* returns PLL_INVALID_FREQ and returns *Param0* as *Result1* since the PLL setting is unchanged.

23.4.1.2 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*.

23.4.1.3 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.

23.4.2 set_power

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

Remark: Use the set_power routine with SYSAHBCLKDIV = 1 (System clock divider register, see Table 17 and Figure 45).

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

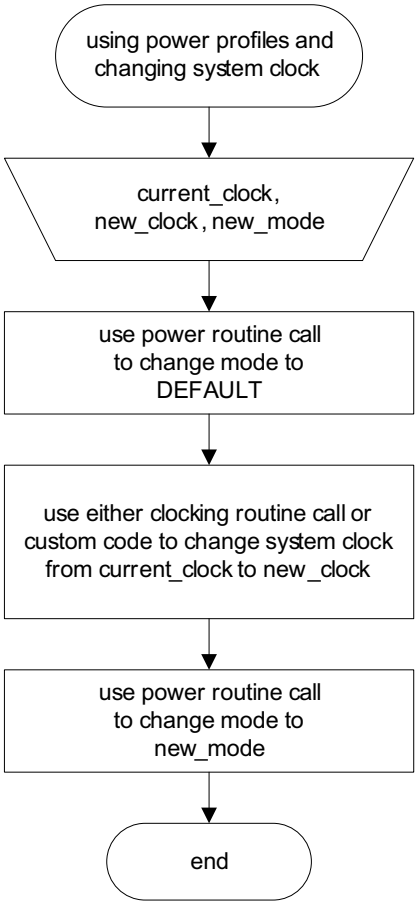


Fig 46. Power profiles usage

Table 257. 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 45. For more details see Figure 3.

23.4.2.1 Param0: main clock

The main clock is the clock rate the microcontroller uses to source the system's and the peripherals' clock. It is configured by either a successful execution of the clocking routine call or a similar code provided by the user. This operand must be an integer between 1 to 30 MHz inclusive. If a value out of this range is supplied, *set_power* returns PWR_INVALID_FREQ and does not change the power control system.

23.4.2.2 Param1: mode

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

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

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

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

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

23.4.2.3 Param2: system clock

The system clock is the clock rate at which the microcontroller core is running when *set_power* is called. This parameter is an integer between from 1 and 30 MHz inclusive.

23.5 Functional description

23.5.1 Clock control

See Section 23.5.1.1 to Section 23.5.1.6 for examples of the clock control API.

23.5.1.1 Invalid frequency (device maximum clock rate exceeded)

```
command[0] = 12000;
command[1] = 60000;
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 60 MHz. The application was ready to infinitely wait for the PLL to lock. But the expected system clock of 60 MHz exceeds the maximum of 30 MHz. Therefore *set_pll* returns PLL_INVALID_FREQ in *result[0]* and 12000 in *result[1]* without changing the PLL settings.

23.5.1.2 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.

23.5.1.3 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.

23.5.1.4 System clock less than or equal to the expected value

```
command[0] = 12000;
command[1] = 25000;
command[2] = CPU_FREQ_LTE;
command[3] = 0;
(*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.

23.5.1.5 System clock greater than or equal to the expected value

```
command[0] = 12000;
command[1] = 20000;
command[2] = CPU_FREQ_GTE;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock, a system clock of at least 20 MHz and no locking time-out. *set_pll* returns PLL_CMD_SUCCESS in *result[0]* and 24000 in *result[1]*. The new system clock is 24 MHz.

23.5.1.6 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.

23.5.2 Power control

See Section 23.5.1.1 and Section 23.5.2.2 for examples of the power control API.

23.5.2.1 Invalid frequency (device maximum clock rate exceeded)

```
command[0] = 30;
command[1] = PWR_CPU_PERFORMANCE;
command[2] = 40;
(*rom)->pWRD->set_power(command, result);
```

The above setup would be used in a system running at the main and system clock of 30 MHz, with a need for maximum CPU processing power. Since the specified 40 MHz clock is above the 30 MHz maximum, *set_power* returns PWR_INVALID_FREQ in *result[0]* without changing anything in the existing power setup.

23.5.2.2 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.

24.1 How to read this chapter

The I2C-bus ROM API is available on all LPC800 parts.

24.2 Features

- Simple I2C drivers to send and receive data on the I2C-bus.
- Polled and interrupt-driven receive and transmit functions for master and slave modes.

24.3 General description

The drivers are callable for use by any application program to send or receive data on the I2C bus. With the I2C drivers it is easy to produce working projects using the I2C interface.

The ROM routines allow the user to operate the I2C interface as a Master or a Slave. The software routines do not implement arbitration to make a Master switch to a Slave mode in the midst of a transmission.

Although multi-master arbitration is not implemented in these I2C drivers, it is possible to use them in a system design with more than one master. If the flag returned from the driver indicates that the message was not successful due to loss of arbitration, the application just resends the message.

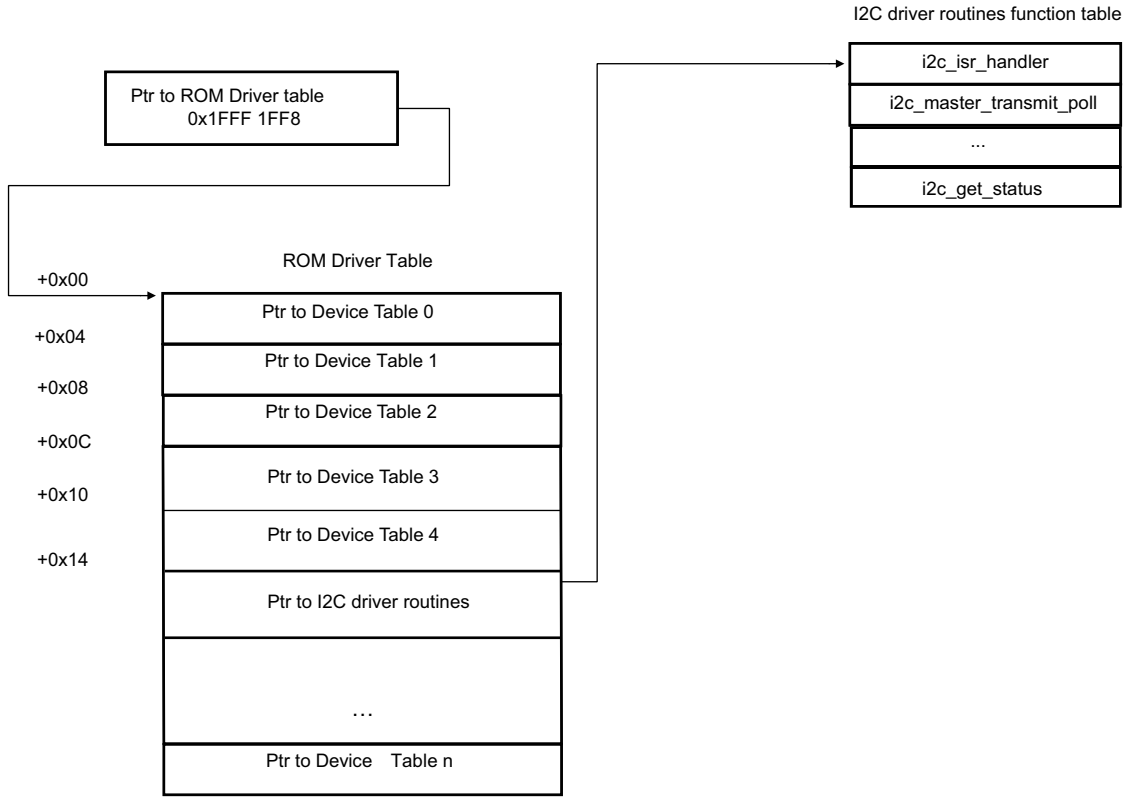


Fig 47. I2C-bus driver routines pointer structure

24.4 API description

The I2C API contains functions to configure the I2C and send and receive data in master and slave modes.

Table 258. I2C API calls

API call	Description	Reference
void i2c_isr_handler(I2C_HANDLE_T*)	I2C ROM Driver interrupt service routine.	Table 259
ErrorCode_t i2c_master_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)	I2C Master Transmit Polling	Table 260
ErrorCode_t i2c_master_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)	I2C Master Receive Polling	Table 261
ErrorCode_t i2c_master_tx_rx_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)	I2C Master Transmit and Receive Polling	Table 262
ErrorCode_t i2c_master_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)	I2C Master Transmit Interrupt	Table 263
ErrorCode_t i2c_master_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)	I2C Master Receive Interrupt	Table 264
ErrorCode_t i2c_master_tx_rx_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)	I2C Master Transmit Receive Interrupt	Table 265

Table 258. I2C API calls

API call	Description	Reference
ErrorCode_t i2c_slave_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)	I2C Slave Receive Polling	Table 266
ErrorCode_t i2c_slave_transmit_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)	I2C Slave Transmit Polling	Table 267
ErrorCode_t i2c_slave_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)	I2C Slave Receive Interrupt	Table 268
ErrorCode_t i2c_slave_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)	I2C Slave Transmit Interrupt	Table 269
ErrorCode_t i2c_set_slave_addr(I2C_HANDLE_T*, slave_addr_0_3, slave_mask_0_3)	I2C Set Slave Address	Table 270
uint32_t i2c_get_mem_size(void)	I2C Get Memory Size	Table 271
I2C_HANDLE_T* i2c_setup(i2c_base_addr, *start_of_ram)	I2C Setup	Table 272
ErrorCode_t i2c_set_bitrate(I2C_HANDLE_T*, P_clk_in_hz, bitrate_in_bps)	I2C Set Bit Rate	Table 273
uint32_t i2c_get_firmware_version(void)	I2C Get Firmware Version	Table 274
I2C_MODE_T i2c_get_status(I2C_HANDLE_T*)	I2C Get Status	Table 275
ErrorCode_t i2c_set_timeout(I2C_HANDLE_T* h_i2c, uint32_t timeout)	I2C time-out value	Table 276

The following structure has to be defined to use the I2C API:

```
typedef struct I2CD_API {   // index of all the i2c driver functions
void (*i2c_isr_handler) (I2C_HANDLE_T* h_i2c) ; // ISR interrupt service request
// MASTER functions ***
ErrorCode_t (*i2c_master_transmit_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
                                                    I2C_RESULT* ptr );
ErrorCode_t (*i2c_master_receive_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
                                                     I2C_RESULT* ptr );
ErrorCode_t (*i2c_master_tx_rx_poll)(I2C_HANDLE_T* h_i2c,I2C_PARAM* ptp,
                                                    I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_master_transmit_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
     I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_master_receive_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
     I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_master_tx_rx_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT*
     ptr ) ;
// SLAVE functions ***
ErrorCode_t (*i2c_slave_receive_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT*
     ptr ) ;
ErrorCode_t (*i2c_slave_transmit_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
     I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_slave_receive_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT*
     ptr ) ;
ErrorCode_t (*i2c_slave_transmit_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
     I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_set_slave_addr)(I2C_HANDLE_T* h_i2c,
                                uint32_t slave_addr_0_3, uint32_t slave_mask_0_3);
// OTHER functions
uint32_t (*i2c_get_mem_size)(void) ; //ramsize_in_bytes memory needed by I2C drivers
```

```
I2C_HANDLE_T*   (*i2c_setup)(uint32_t i2c_base_addr, uint32_t *start_of_ram ) ;
ErrorCode_t   (*i2c_set_bitrate)(I2C_HANDLE_T* h_i2c, uint32_t P_clk_in_hz,
                                 uint32_t bitrate_in_bps) ;
uint32_t (*i2c_get_firmware_version)() ;
I2C_MODE_T (*i2c_get_status)(I2C_HANDLE_T* h_i2c ) ;
} I2CD_API_T ;
```

24.4.1 ISR handler

Table 259. ISR handler

Routine	ISR handler
Prototype	void i2c_isr_handler(I2C_HANDLE_T*)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area.
Return	None.
Description	I2C ROM Driver interrupt service routine. This function must be called from the I2C ISR when using I2C Rom Driver interrupt mode.

24.4.2 I2C Master Transmit Polling

Table 260. I2C Master Transmit Polling

Routine	I2C Master Transmit Polling
Prototype	ErrorCode_t i2c_master_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Transmits bytes in the send buffer to a slave. The slave address with the R/W bit =0 is expected in the first byte of the send buffer. STOP condition is sent at end unless stop_flag =0. When the task is completed, the function returns to the line after the call.

24.4.3 I2C Master Receive Polling

Table 261. I2C Master Receive Polling

Routine	I2C Master Receive Polling
Prototype	ErrorCode_t i2c_master_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Receives bytes from slave and put into receive buffer. The slave address with the R/W bit =0 is expected in the first byte of the send buffer. After the task is finished, the slave address with the R/W bit =1 is in the first byte of the receive buffer. STOP condition is sent at end unless stop_flag =0. When the task is completed, the function returns to the line after the call.

24.4.4 I2C Master Transmit and Receive Polling

Table 262. I2C Master Transmit and Receive Polling

Routine	I2C Master Transmit and Receive Polling
Prototype	ErrorCode_t i2c_master_tx_rx_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	First, transmit bytes in the send buffer to a slave and secondly, receives bytes from slave and store it in the receive buffer. The slave address with the R/W bit =0 is expected in the first byte of the send buffer. After the task is finished, the slave address with the R/W bit =1 is in the first byte of the receive buffer. STOP condition is sent at end unless stop_flag =0. When the task is completed, the function returns to the line after the call.

24.4.5 I2C Master Transmit Interrupt

Table 263. I2C Master Transmit Interrupt

Routine	I2C Master Transmit Interrupt
Prototype	ErrorCode_t i2c_master_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Transmits bytes in the send buffer to a slave. The slave address with the R/W bit =0 is expected in the first byte of the send buffer. STOP condition is sent at end unless stop_flag =0. Program control will be returned immediately and task will be completed on an interrupt-driven basis. When task is completed, the callback function is called.

24.4.6 I2C Master Receive Interrupt

Table 264. I2C Master Receive Interrupt

Routine	I2C Master Receive Interrupt
Prototype	ErrorCode_t i2c_master_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Receives bytes from slave and put into receive buffer. After the task is finished, the slave address with the R/W bit =1 is in the first byte of the receive buffer. STOP condition is sent at end unless stop_flag =0. Program control will be returned immediately and task will be completed on an interrupt-driven basis. When task is completed, the callback function is called.

24.4.7 I2C Master Transmit Receive Interrupt

Table 265. I2C Master Transmit Receive Interrupt

Routine	I2C Master Transmit Receive Interrupt
Prototype	ErrorCode_t i2c_master_tx_rx_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	First, transmits bytes in the send buffer to a slave and secondly, receives bytes from slave and store it in the receive buffer. The slave address with the R/W bit =0 is expected in the first byte of the send buffer. After the task is finished, the slave address with the R/W bit =1 is in the first byte of the receive buffer. STOP condition is sent at end unless stop_flag =0. Program control will be returned immediately and task will be completed on an interrupt-driven basis. When task is completed, the callback function is called.

24.4.8 I2C Slave Receive Polling

Table 266. I2C Slave Receive Polling

Routine	I2C Slave Receive Polling
Prototype	ErrorCode_t i2c_slave_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Receives data from master. When the task is completed, the function returns to the line after the call.

24.4.9 I2C Slave Transmit Polling

Table 267. I2C Slave Transmit Polling

Routine	I2C Slave Transmit Polling
Prototype	ErrorCode_t i2c_slave_transmit_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Sends data bytes back to master. When the task is completed, the function returns to the line after the call.

24.4.10 I2C Slave Receive Interrupt

Table 268. I2C Slave Receive Interrupt

Routine	I2C Slave Receive Interrupt
Prototype	ErrorCode_t i2c_slave_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Receives data from master. Program control will be returned immediately and task will be completed on an interrupt-driven basis. When task is completed, the callback function is called.

24.4.11 I2C Slave Transmit Interrupt

Table 269. I2C Slave Transmit Interrupt

Routine	I2C Slave Transmit Interrupt
Prototype	ErrorCode_t i2c_slave_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Sends data to the Master. Program control will be returned immediately and task will be completed on an interrupt-driven basis. When task is completed, the callback function is called.

24.4.12 I2C Set Slave Address

Table 270. I2C Set Slave Address

Routine	I2C Set Slave Address
Prototype	ErrorCode_t i2c_set_slave_addr(I2C_HANDLE_T*, slave_addr_0_3, slave_mask_0_3)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. Slave_addr_0_3 - unint32 variable. 7-bit slave address . Slave_mask_0_3 - unint32 variable. Slave address mask.
Return	ErrorCode.
Description	Sets the slave address and associated mask. The set_slave_addr() function supports four 7-bit slave addresses and masks.

24.4.13 I2C Get Memory Size

Table 271. I2C Get Memory Size

Routine	I2C Get Memory Size
Prototype	uint32_t i2c_get_mem_size(void)

Table 271. I2C Get Memory Size

Routine	I2C Get Memory Size
Input parameter	None.
Return	uint32.
Description	Returns the number of bytes in SRAM needed by the I2C driver.

24.4.14 I2C Setup

Table 272. I2C Setup

Routine	I2C Setup
Prototype	I2C_HANDLE_T* i2c_setup(i2c_base_addr, *start_of_ram)
Input parameter	I2C_base addr - unint32 variable. Base address for I2C peripherals. Start_of_ram - unint32 pointer. Pointer to allocated SRAM.
Return	I2C_Handle.
Description	Returns a handle to the allocated SRAM area.

24.4.15 I2C Set Bit Rate

Table 273. I2C Set Bit Rate

Routine	I2C Set Bit Rate
Prototype	ErrorCode_t i2c_set_bitrate(I2C_HANDLE_T*, P_clk_in_hz, bitrate_in_bps)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. P_clk_in_hz - unint32 variable. The Peripheral Clock in Hz. Bitrate_in_bps - unint32 variable. Requested I2C operating frequency in Hz.
Return	ErrorCode.
Description	Configures the I2C duty-cycle registers (SCLH and SCLL).

24.4.16 I2C Get Firmware Version

Table 274. I2C Get Firmware Version

Routine	I2C Get Firmware Version
Prototype	uint32_t i2c_get_firmware_version(void)
Input parameter	None.
Return	I2C ROM Driver version number.
Description	Returns the version number. The firmware version is an unsigned 32-bit number.

24.4.17 I2C Get Status

Table 275. I2C Get Status

Routine	I2C Get Status
Prototype	I2C_MODE_T i2c_get_status(I2C_HANDLE_T*)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area.
Return	Status code.
Description	Returns status code. The status code indicates the state of the I2C bus. Refer to I2C Status Code Table.

24.4.18 I2C time-out value

Table 276. I2C time-out value

Routine	I2C time-out value
Prototype	ErrorCode_t i2c_set_timeout(I2C_HANDLE_T* h_i2c, uint32_t timeout)
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. uint32_t timeout - time value is timeout*16 i2c function clock. If timeout = 0, timeout feature is disabled.
Return	Status code.
Description	Returns status code. The status code indicates the state of the I2C bus. Refer to I2C Status Code Table.

24.4.19 Error codes

Table 277. Error codes

Error Code	Description	Comment
0	Successful completion	Function was completed successfully.
1	General error	-
0x0006 0001	ERR_I2C_NAK	-
0x0006 0002	ERR_I2C_BUFFER_OVERFLOW	-
0x0006 0003	ERR_I2C_BYTE_COUNT_ERR	-
0x0006 0004	ERR_I2C_LOSS_OF_ARBRITRATION	-
0x0006 0005	ERR_I2C_SLAVE_NOT_ADDRESSED	-
0x0006 0006	ERR_I2C_LOSS_OF_ARBRITRATION_NAK_BIT	-
0x0006 0007	ERR_I2C_GENERAL_FAILURE	Failure detected on I2C bus.
0x0006 0008	ERR_I2C_REGS_SET_TO_DEFAULT	I2C clock frequency could not be set. Default value of 0x04 is loaded into SCLH and SCLL.

24.4.20 I2C Status code

Table 278. I2C Status code

Status code	Description
0	IDLE
1	MASTER_SEND
2	MASTER_RECEIVE
3	SLAVE_SEND
4	SLAVE_RECEIVE

24.4.21 I2C ROM driver variables

The I2C ROM driver requires specific variables to be declared and initialized for proper usage. Depending on the operating mode, some variables can be omitted.

24.4.21.1 I2C Handle

The I2C handle is a pointer allocated for the I2C ROM driver. The handle needs to be defined as an I2C handle TYPE:

```
typedef void* I2C_HANDLE_T
```

After the definition of the handle, the handle must be initialized with I2C base address and RAM reserved for the I2C ROM driver by making a call to the i2c_setup() function.

The callback function type must be defined if interrupts for the I2C ROM driver are used:

```
typedef void (*I2C_CALLBK_T) (uint32_t err_code, uint32_t n)
```

The callback function will be called by the I2C ROM driver upon completion of a task when interrupts are used.

24.4.22 PARAM and RESULT structure

The I2C ROM driver input parameters consist of two structures, a PARAM structure and a RESULT structure. The PARAM structure contains the parameters passed to the I2C ROM driver and the RESULT structure contains the results after the I2C ROM driver is called.

The PARAM structure is as follows:

```
typedef struct i2c_A { //parameters passed to ROM function
     uint32_t num_bytes_send ;
     uint32_t num_bytes_rec ;
     uint8_t *buffer_ptr_send ;
     uint8_t *buffer_ptr_rec ;
     I2C_CALLBK_T func_pt; // callback function pointer
     uint8_t stop_flag;
     uint8_t dummy[3] ;         // required for word alignment
} I2C_PARAM ;
```

The RESULT structure is as follows:

```
typedef struct i2c_R {     // RESULTs struct--results are here when returned
     uint32_t n_bytes_sent ;
     uint32_t n_bytes_recd ;
} I2C_RESULT ;
```

24.4.23 Error structure

The error code returned by the I2C ROM driver is an enum structure. The Error structure is as follows:

```
typedef enum
{
                LPC_OK=0, /**< enum value returned on Success */
                 ERROR,
 ERR_I2C_BASE = 0x00060000,
/*0x00060001*/ ERR_I2C_NAK=ERR_I2C_BASE+1,
/*0x00060002*/ ERR_I2C_BUFFER_OVERFLOW,
/*0x00060003*/ ERR_I2C_BYTE_COUNT_ERR,
/*0x00060004*/ ERR_I2C_LOSS_OF_ARBRITRATION,
/*0x00060005*/ ERR_I2C_SLAVE_NOT_ADDRESSED,
/*0x00060006*/ ERR_I2C_LOSS_OF_ARBRITRATION_NAK_BIT,
/*0x00060007*/ ERR_I2C_GENERAL_FAILURE,
/*0x00060008*/ ERR_I2C_REGS_SET_TO_DEFAULT
```

```
} ErrorCode_t;
```

24.4.24 I2C Mode

The i2c_get_status() function returns the current status of the I2C engine. The return codes can be defined as an enum structure:

```
typedef enum I2C_mode {
 IDLE,
 MASTER_SEND,
 MASTER_RECEIVE,
 SLAVE_SEND,
 SLAVE_RECEIVE
 } I2C_MODE_T ;
```

24.4.25 I2C ROM driver pointer

The I2C ROM driver resides in the address 0x1FFF1FF8. The address must be declared to allow access to the ROM driver:

```
#define ROM_DRIVERS_PTR ((ROM *)(*((unsigned int *)0x1FFF1FF8)))
```

24.5 Functional description

24.5.1 I2C Set-up

Before calling any setup functions in the I2C ROM, the application program is responsible for doing the following:

1. Enable the clock to the I2C peripheral.
2. Enable the two pins required for the SCL and SDA outputs of the I2C peripheral.
3. Allocate a RAM area for dedicated use of the I2C ROM Driver.

After the I2C block is configured, the I2C ROM driver variables have to be set up:

1. Initialize pointer to the I2C API function table.
2. Declare the PARAM and RESULT struct.
3. Declare Error Code struct.
4. Declare the transmit and receive buffer.

If interrupts are used, then additional driver variables have to be set up:

1. Declare the I2C_CALLBK_T type.
2. Declare callback functions.
3. Declare I2C ROM Driver ISR within the I2C ISR.
4. Enable I2C interrupt.

24.5.2 I2C Master mode set-up

The I2C ROM Driver support polling and interrupts. In the master mode, 7-bit and 10-bit addressing are supported. The setup is as follows:

1. Allocate SRAM for the I2C ROM Driver by making a call to the i2c_get_mem_size() function.
2. Create the I2C handle by making a call to the i2c_setup() function.
3. Set the I2C operating frequency by making a call to the i2c_set_bitrate() function.

```
pI2cApi = ROM_DRIVERS_PTR->pI2CD; //setup I2C function table pointer
size_in_bytes = pI2cApi->i2c_get_mem_size();
i2c_handle = pI2cApi->i2c_setup(LPC_I2C_BASE, (uint32_t *)&I2C_Handle[0] );
error_code = pI2cApi->i2c_set_bitrate((I2C_HANDLE_T*)i2c_handle, PCLK_in_Hz,
     bps_in_hz);
```

24.5.3 I2C Slave mode set-up

The I2C ROM Driver support polling and interrupts in the slave mode. In the slave mode, only 7-bit addressing is supported. The set-up is as follows:

1. Allocate SRAM for the I2C ROM Driver by making a call to the i2c_get_mem_size() function.
2. Create the I2C handle by making a call to the i2c_setup() function.
3. Set the I2C operating frequency by making a call to the i2c_set_bitrate() function.
4. Set the slave address by making a call to the i2c_set_slave_addr() function.

The I2C ROM driver allows setting up to 4 slave addresses and 4 address masks as well as possibly enabling the General Call address.

The four slave address bytes are packed into the 4 byte variable. Slave address byte 0 is the least significant byte and Slave address byte 3 is the most significant byte. The Slave address mask bytes are ordered the same way in the other 32 bit variable. When in slave receive mode, all of these addresses (or groups if masks are used) will be monitored for a match. If the General Call bit (least significant bit of any of the four slave address bytes) is set, then the General Call address of 0x00 is monitored as well.

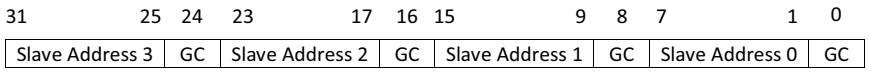


Fig 48. I2C slave mode set-up address packing

```
pI2cApi = ROM_DRIVERS_PTR->pI2CD; //setup I2C function table pointer
size_in_bytes = pI2cApi->i2c_get_mem_size();
i2c_handle = pI2cApi->i2c_setup(LPC_I2C_BASE, (uint32_t *)&I2C_Handle[0] );
error_code = pI2cApi->i2c_set_bitrate((I2C_HANDLE_T*)i2c_handle, PCLK_in_Hz,
     bps_in_hz);
error_code = pI2cApi->i2c_set_slave_addr((I2C_HANDLE_T*)i2c_handle, slave_addr,
     slave_addr_mask) ;
```

24.5.4 I2C Master Transmit/Receive

The Master mode drivers give the user the choice of either polled (wait for the message to finish) or interrupt driven routines (non-blocking). Polled routines are recommended for testing purposes or very simple I2C applications. These routines allow the Master to send to Slaves with 7-bit or 10-bit addresses.

The following routines are polled routines :

```
err_code i2c_master_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_master_receive_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_master_tx_rx_poll (I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

The following routines are interrupt driven routines:

```
err_code i2c_master_transmit_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_master_receive_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_master_tx_rx_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

Where:

- err_code is the return state of the function. An “0” indicates success. All non-zero indicates an error. Refer to Error Table.
- I2C_PARM* is a structure with parameters passed to the function. Refer to Section 24.4.22.
- I2C_RESULT* is a containing the results after the function executes.

To initiate a master mode write/read the I2C_PARAM has to be setup. The I2C_PARAM is a structure with various variables needed by the I2C ROM Driver to operate correctly. The structure contains the following:

- Number of bytes to be transmitted.
- Number of bytes to be receive.
- Pointer to the transmit buffer.
- Pointer to the receive buffer.
- Pointer to callback function.
- Stop flag.

The RESULT structure contains the results after the function executes. The structure contains the following:

- Number of bytes transmitted.
- Number of bytes received.

Remark: The number of bytes transmitted will be updated for i2c_master_transmit_intr() and i2c_master_transmit_poll(). The number of bytes received will only be update on i2c_master_receive_poll(), i2c_master_receive_intr(), i2c_master_tx_rx_poll(), and i2c_master_tx_rx_intr().

In all the master mode routines, the transmit buffer's first byte must be the slave address with the R/W bit set to "0". To enable a master read, the receive buffer's first byte must be the slave address with the R/W bit set to "1".

The following conditions must be fulfilled to use the I2C driver routines in master mode:

- For 7-bit addressing, the first byte of the send buffer must have the slave address in the most significant 7 bits and the least significant (R/W) bit = 0. Example: Slave address 0x53, first byte is 0xA6.
- For 7-bit addressing, the first byte of the receive buffer must have the slave address in the most significant 7 bits and the least significant (R/W) bit = 1. Example: Slave Addr 0x53, first byte 0xA7.
- For 10-bit address, the first byte of the transmit buffer must have the slave address most significant 2 bits with the (R/W) bit =0. The second byte must contain the remaining 8-bit of the slave address.
- For 10-bit address, the first byte of the receive buffer must have the slave address most significant 2 bits with the (R/W) bit =1. The second byte must contain the remaining 8-bit of the slave address.
- The number of bytes to be transmitted should include the first byte of the buffer which is the slave address byte. Example: 2 data bytes + 7-bit slave addr = 3.
- The application program must enable I2C interrupts. When I2C interrupt occurs, the i2c_isr_handler function must be called from the application program.

When using the interrupt function calls, the callback functions must be define. Upon the completion of a read/write as specified by the PARAM structure, the callback functions will be invoked.

24.5.5 I2C Slave Mode Transmit/Receive

In slave mode, polled routines are intended for testing purposes. It is up to the user to decide whether to use the polled or interrupt driven mode. While operating the Slave driver in polled mode can be useful for program development and debugging, most applications will need the interrupt-driven versions of Slave Receive and Transmit in the final software.

The following routines are polled routines:

```
err_code i2c_slave_receive_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_slave_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

The following routines are interrupt driven routines:

```
err_code i2c_slave_receive_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_slave_transmit_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

Where:

- err_code is the return state of the function. An 0 indicates success. All non-zero indicates an error. Refer to the Error Code Table.
- I2C_PARM is a structure with parameters passed to the function. Section 24.4.22.

- I2C_RESULT is a containing the results after the function executes. Section 24.4.22.

To initiate a master-mode write/read the I2C_PARAM has to be setup. The I2C_PARAM is a structure with various variables needed by the I2C ROM Driver to operate correctly. The structure contains the following:

- Number of bytes to be transmitted.
- Number of bytes to be received.
- Pointer to the transmit buffer.
- Pointer to the receive buffer.
- Pointer to callback function.
- Stop flag.

The RESULT structure contains the results after the function executes. The structure contains the following:

- Number of bytes transmitted.
- Number of bytes received.

Remark: The number of bytes transmitted is updated only for i2c_slave_send_poll() and i2c_slave_send_intr(). The number of bytes received is updated only for i2c_slave_receive_poll() and i2c_slave_receive_intr().

To initiate a slave mode communication, the receive function is called. This can be either the polling or interrupt driven function, i2c_slave_receive_poll() or i2c_slave_receive_intr(), respectively. The receive buffer should be as large or larger than any data or command that will be received. If the amount of data exceed the receive buffer size, an error code will be returned.

In slave-receive mode, the driver receives data until one of the following are true:

- Address matching set in the set_slave_addr() function with the R/W bit set to 1
- STOP or repeated START is received
- An error condition is detected

When using the interrupt function calls, the callback functions must be define. Upon the completion of a read/write as specified by the PARAM structure, the callback functions will be invoked.

24.5.6 I2C time-out feature

```
//timeout: Timeout time value. Specifies the timeout interval value in increments of
// 16 I2C function clocks (Min value is 16).
//                    if timeout = 0,  timeout feature is disabled
//         if timeout != 0, time value is timeout*16 i2c function clock.
ErrorCode_t i2c_set_timeout(I2C_HANDLE_T*  h_i2c, uint32_t timeout)
{
  I2C_DRIVER_TypeDef *h ;  // declare pointer to i2c structure [handle]
  h = (I2C_DRIVER_TypeDef*) h_i2c ;   //assign handle pointer address
     if (timeout != 0){
          h->i2c_base->TimeOut = (timeout - 1)<<4;
```

```
            // Enable timeout feature
            h->i2c_base->CFG |= BI2C_TIMEOUT_EN;
        }
        else
            // disable timeout feature
            h->i2c_base->CFG &= ~BI2C_TIMEOUT_EN;

        return(LPC_OK) ;
}//i2c_set_timeout
```

25.1 How to read this chapter

The USART ROM driver routines are available on all LPC800 parts.

25.2 Features

- Send and receive characters in asynchronous or synchronous mode
- Send and receive multiple characters (line) in asynchronous or synchronous UART mode

25.3 General description

The UART API handles sending and receiving characters using any of the USART blocks in asynchronous mode.

Remark: Because all USARTS share a common fractional divider, the uart_init routine returns the value for the common divider.

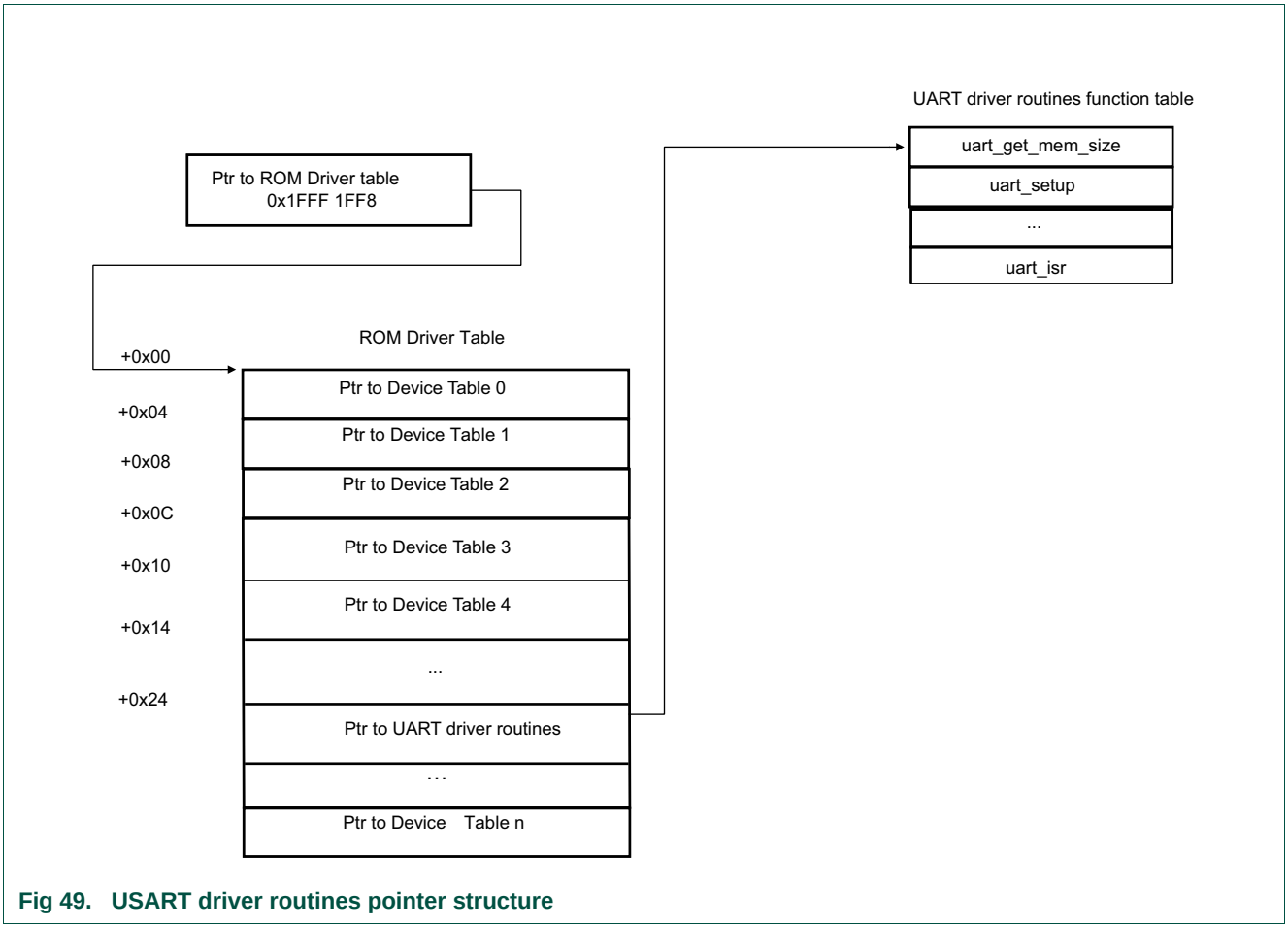


Fig 49. USART driver routines pointer structure

25.4 API description

The UART API contains functions to send and receive characters via any of the USART blocks.

Table 279. UART API calls

API call	Description	Reference
uint32_t ramsize_in_bytes uart_get_mem_size(void) ;	UART get memory size	Table 280
UART_HANDLE_T* uart_setup(uint32_t base_addr, uint8_t *ram) ;	UART set-up	Table 281
uint32_t uart_init(UART_HANDLE_T* handle, UART_CONFIG set);	UART init	Table 282
uint8_t uart_get_char(UART_HANDLE_T* handle);	UART get character	Table 283
void uart_put_char(UART_HANDLE_T* handle, uint8_t data);	UART put character	Table 284
uint32_t uart_get_line(UART_HANDLE_T* handle, UART_PARAM_T param);	UART get line	Table 285
uint32_t uart_put_line(UART_HANDLE_T* handle, UART_PARAM_T param);	UART put line	Table 286
void uart_isr(UART_HANDLE_T* handle);	UART interrupt service routine	Table 287

The following structure has to be defined to use the UART API:

```
typedef struct  UARTD_API {         // index of all the uart driver functions
            uint32_t (*uart_get_mem_size)(void);
            UART_HANDLE_T (*uart_setup)(uint32_t base_addr, uint8_t *ram);
            uint32_t (*uart_init)(UART_HANDLE_T handle, UART_CONFIG_T *set);
            //--polling functions--//
            uint8_t (*uart_get_char)(UART_HANDLE_T handle);
            void (*uart_put_char)(UART_HANDLE_T handle, uint8_t data);
            uint32_t (*uart_get_line)(UART_HANDLE_T handle, UART_PARAM_T * param);
            uint32_t (*uart_put_line)(UART_HANDLE_T handle, UART_PARAM_T * param);
            //--interrupt functions--//
            void (*uart_isr)(UART_HANDLE_T handle);
} UARTD_API_T  ;            // end of structure
```

25.4.1 UART get memory size

Table 280. uart_get_mem_size

Routine	uart_get_mem_size
Prototype	uint32_t ramsize_in_bytes uart_get_mem_size(void) ;
Input parameter	None.
Return	Memory size in bytes.
Description	Get the memory size needed by one UART instance.

25.4.2 UART setup

Table 281. uart_setup

Routine	uart_setup
Prototype	UART_HANDLE_T* uart_setup(uint32_t base_addr, uint8_t *ram) ;
Input parameter	base_addr: Base address of register for this uart block. ram: Pointer to the memory space for uart instance. The size of the memory space can be obtained by the uart_get_mem_size function.
Return	The handle to corresponding uart instance.
Description	Setup UART instance with provided memory and return the handle to this instance.

25.4.3 UART init

Table 282. uart_init

Routine	uart_init
Prototype	uint32_t uart_init(UART_HANDLE_T* handle, UART_CONFIG set);
Input parameter	handle: The handle to the uart instance. set: configuration for uart operation.
Return	Fractional divider value if System clock is not integer multiples of baud rate.
Description	Setup baud rate and operation mode for uart, then enable uart.

25.4.4 UART get character

Table 283. uart_get_char

Routine	uart_get_char
Prototype	uint8_t uart_get_char(UART_HANDLE_T* handle);
Input parameter	handle: The handle to the uart instance.
Return	Received data
Description	Receive one Char from uart. This functions is only returned after Char is received. In case Echo is enabled, the received data is sent out immediately.

25.4.5 UART put character

Table 284. uart_put_char

Routine	uart_put_char
Prototype	void uart_put_char(UART_HANDLE_T* handle, uint8_t data);
Input parameter	handle: The handle to the uart instance. data: data to be sent out.
Return	None.
Description	Send one Char through uart. This function is only returned after data is sent.

25.4.6 UART get line

Table 285. uart_get_line

Routine	uart_get_line
Prototype	uint32_t uart_get_line(UART_HANDLE_T* handle, UART_PARAM_T param);
Input parameter	handle: The handle to the uart instance. param: Refer to UART_PARAM_T definition.
Return	Error code: ERR_UART_RECEIVE_ON - UART receive is ongoing.
Description	Receive multiple bytes from UART.

25.4.7 UART put line

Table 286. uart_put_line

Routine	uart_put_line
Prototype	uint32_t uart_put_line(UART_HANDLE_T* handle, UART_PARAM_T param);
Input parameter	handle: The handle to the uart instance. param: Refer to UART_PARAM_T definition.
Return	Error code: ERR_UART_SEND_ON - UART sending is ongoing.
Description	Send string (end with \0) or raw data through UART.

25.4.8 UART interrupt service routine

Table 287. uart_isr

Routine	uart_isr
Prototype	void uart_isr(UART_HANDLE_T* handle);
Input parameter	handle: The handle to the uart instance.
Return	None.
Description	UART interrupt service routine. To use this routine, the corresponding USART interrupt must be enabled. This function is invoked by the user ISR.

25.4.9 Error codes

Table 288. Error codes

Return code	Error Code	Description
0x0008 0001	ERR_UART_RXD_BUSY = ERR_UART_BASE+1,	UART receive is busy
0x0008 0002	ERR_UART_TXD_BUSY	UART transmit is busy
0x0008 0003	ERR_UART_OVERRUN_FRAME_PARITY_NOISE	Overrun error, Frame error, parity error, RxNoise error
0x0008 0004	ERR_UART_UNDERRUN	Underrun error
0x0008 0005	ERR_UART_PARAM	Parameter error

25.4.10 UART ROM driver variables

25.4.10.1 UART_CONFIG structure

```
typdef  struct UART_CONFIG {
        uint32_t sys_clk_in_hz; // Sytem clock in hz.
        uint32_t baudrate_in_hz; // Baudrate in hz
        uint8_t  config; //bit 1:0
                        // 00: 7 bits length, 01: 8 bits lenght, others: reserved
                        //bit3:2
                        //   00: No Parity, 01: reserved, 10: Even, 11: Odd
                        //bit4
                        //   0: 1 Stop bit, 1: 2 Stop bits
        uint8_t sync_mod;  //bit0: 0(Async mode), 1(Sync mode)
                                //bit1: 0(Un_RXD is sampled on the falling edge of SCLK)
                                //       1(Un_RXD is sampled on the rising edge of SCLK)
                                //bit2: 0(Start and stop bits are transmitted as in asynchronous   mode)
                                //       1(Start and stop bits are not transmitted)
                                //bit3: 0(the UART is a slave on Sync mode)
                                //       1(the UART is a master on Sync mode)
uint16_t error_en; //Bit0: OverrunEn, bit1: UnderrunEn, bit2: FrameErrEn,
                        // bit3: ParityErrEn, bit4: RxNoiseEn
}
```

25.4.10.2 UART_HANDLE_T

The handle to the instance of the UART driver. Each UART has one handle, so there can be several handles for up to three UART blocks. This handle is created by Init API and used by the transfer functions for the corresponding UART block.

```
typedef  void    *UART_HANDLE_T   ;    // define TYPE for uart handle pointer
```

25.4.10.3 UART_PARAM_T

```
typedef struct  uart_A {  // parms passed to uart driver function
uint8_t  *  buffer ; // The pointer of buffer.
                                // For uart_get_line function, buffer for receiving data.
                                // For uart_put_line function, buffer for transmitting data.
uint32_t   size;     // [IN] The size of buffer.
                                //[OUT] The number of bytes transmitted/received.
uint16_t    transfer_mode ;
                                // 0x00:  For uart_get_line function, transfer without
                                // termination.
                                // For uart_put_line function, transfer without termination.
                                // 0x01:  For uart_get_line function, stop transfer when
                                // <CR><LF> are received.
                                // For uart_put_line function, transfer is stopped after
                                // reaching \0. <CR><LF> characters are sent out after that.
                                // 0x02:  For uart_get_line function, stop transfer when <LF>
                                // is received.
                                // For uart_put_line function, transfer is stopped after
                                // reaching \0. A <LF> character is sent out after that.
                                //0x03:  For uart_get_line function, RESERVED.
```

```
                                // For uart_put_line function, transfer is stopped after
                                // reaching \0.
uint16_t   driver_mode;
                                //0x00: Polling mode, function is blocked until transfer is
                                // finished.
                                // 0x01: Intr mode, function exit immediately, callback function
                                // is invoked when transfer is finished.
                                //0x02: RESERVED
UART_CALLBK_T    callback_func_pt; // callback function
} UART_PARAM_T ;
```

26.1 How to read this chapter

The debug functionality is identical for all LPC800 parts.

26.2 Features

- Supports ARM Serial Wire Debug mode.
- Direct debug access to all memories, registers, and peripherals.
- No target resources are required for the debugging session.
- Four breakpoints.
- Two data watchpoints that can also be used as triggers.
- Supports JTAG boundary scan.
- Micro Trace Buffer (MTB) supported.

26.3 General description

Debug functions are integrated into the ARM Cortex-M0+. Serial wire debug functions are supported. The ARM Cortex-M0+ is configured to support up to four breakpoints and two watchpoints.

Support for boundary scan and Micro Trace Buffer is available.

26.4 Pin description

The SWD functions are assigned to pins through the switch matrix. The SWD functions are fixed-pin functions that are enabled through the switch matrix and can only be assigned to special pins on the package. The SWD functions are enabled by default.

See Section 9.3.2 to enable the analog comparator inputs and the reference voltage input.

Table 289. SWD pin description

Function	Type	Pin	Description	SWM register	Reference
SWCLK	I/O	SWCLK/PIO0_3/ TCLK	**Serial Wire Clock.** This pin is the clock for SWD debug logic when in the Serial Wire Debug mode (SWD). This pin is pulled up internally.	PINENABLE0	Table 106
SWDIO	I/O	SWDIO/PIO0_2/ TMS	**Serial wire debug data input/output.** The SWDIO pin is used by an external debug tool to communicate with and control the LPC800. This pin is pulled up internally.	PINENABLE0	Table 106

The boundary scan mode and the pins needed are selected by hardware (see Section 26.5.3). There is no access to the boundary scan pins through the switch matrix.

Table 290. JTAG boundary scan pin description

Function	Pin name	Type	Description
TCK	SWCLK/PIO0_3/ TCK	I	**JTAG Test Clock.** This pin is the clock for JTAG boundary scan when the RESET pin is LOW.
TMS	SWDIO/PIO0_2/ TMS	I	**JTAG Test Mode Select.** The TMS pin selects the next state in the TAP state machine. This pin includes an internal pull-up and is used for JTAG boundary scan when the RESET pin is LOW.
TDI	PIO0_1/ACMP_I2/ CLKIN/TDI	I	**JTAG Test Data In.** This is the serial data input for the shift register. This pin includes an internal pull-up and is used for JTAG boundary scan when the RESET pin is LOW.
TDO	PIO0_0/ACMP_I1/ TDO	O	**JTAG Test Data Output.** This is the serial data output from the shift register. Data is shifted out of the device on the negative edge of the TCK signal. This pin is used for JTAG boundary scan when the RESET pin is LOW.
TRST	PIO0_4/ WAKEUP/TRST	I	**JTAG Test Reset.** The TRST pin can be used to reset the test logic within the debug logic. This pin includes an internal pull-up and is used for JTAG boundary scan when the RESET pin is LOW.

26.5 Functional description

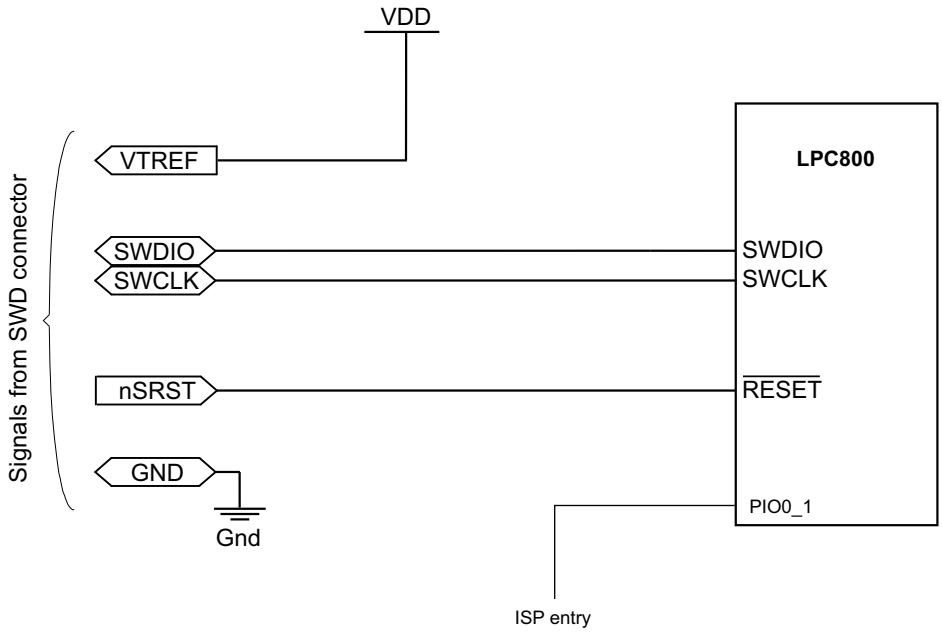
26.5.1 Debug limitations

It is recommended not to use the debug mode during Deep-sleep or Power-down mode mode.

During a debugging session, the System Tick Timer is automatically stopped whenever the CPU is stopped. Other peripherals are not affected.

26.5.2 Debug connections for SWD

For debugging purposes, it is useful to provide access to the ISP entry pin PIO0_1. This pin can be used to recover the part from configurations which would disable the SWD port such as improper PLL configuration, reconfiguration of SWD pins, entry into Deep power-down mode out of reset, etc. This pin can be used for other functions such as GPIO, but it should not be held LOW on power-up or reset.



The VTREF pin on the SWD connector enables the debug connector to match the target voltage.

Fig 50. Connecting the SWD pins to a standard SWD connector

26.5.3 Boundary scan

The $\overline{\text{RESET}}$ pin selects between the JTAG boundary scan ($\overline{\text{RESET}}$ = LOW) and the ARM SWD debug ($\overline{\text{RESET}}$ = HIGH). The ARM SWD debug port is disabled while the part is in reset.

To perform boundary scan testing, follow these steps:

1. Erase any user code residing in flash.
2. Power up the part with the $\overline{\text{RESET}}$ pin pulled HIGH externally.
3. Wait for at least 250 μs.
4. Pull the $\overline{\text{RESET}}$ pin LOW externally.
5. Perform boundary scan operations.
6. Once the boundary scan operations are completed, assert the TRST pin to enable the SWD debug mode and release the $\overline{\text{RESET}}$ pin (pull HIGH).

Remark: The JTAG interface cannot be used for debug purposes.

Remark: POR, BOD reset, or a LOW on the TRST pin puts the test TAP controller in the Test-Logic Reset state. The first TCK clock while $\overline{\text{RESET}}$ = HIGH places the test TAP in Run-Test Idle mode.

26.5.4 Micro Trace Buffer (MTB)

The MTB registers are located at memory address 0x1400 0000 and are described in Ref. 2. The EXTTRACE register in the syscon block (see Section 4.6.20) starts and stops tracing in conjunction with the TSTARTEN and TSTOPEN bits in the MTB MASTER register. The trace is stored in the local SRAM starting at address 0x1000 0000. The trace memory location is configured in the MTB POSITION register.

Remark: The MTB BASE register is not implemented. Reading the BASE register returns 0x0 independently of the SRAM memory area configured for trace.

27.1 Packages

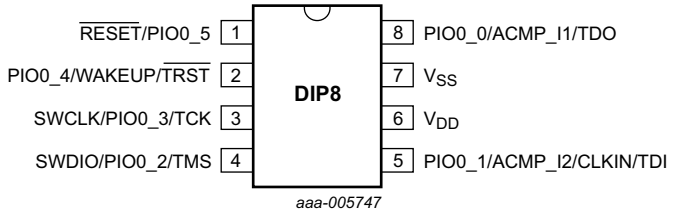


Fig 51. Pin configuration DIP8 package (LPC810M021FN8)

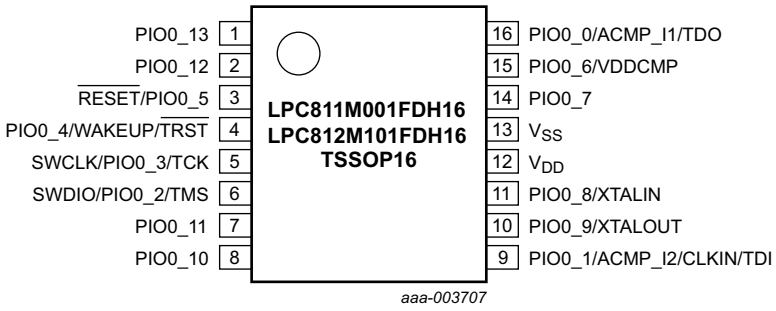


Fig 52. Pin configuration TSSOP16 package

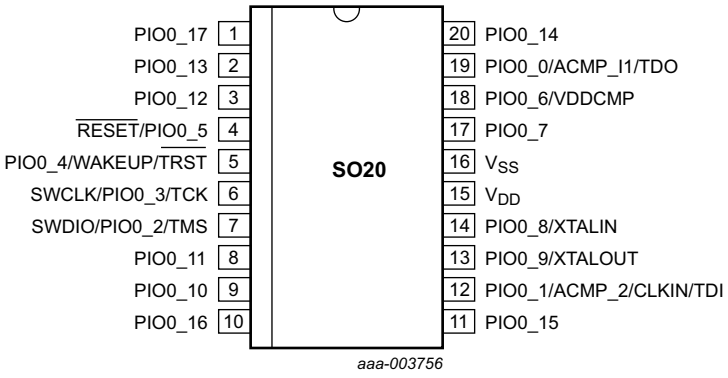


Fig 53. Pin configuration SO20 package (LPC812M101FD20)

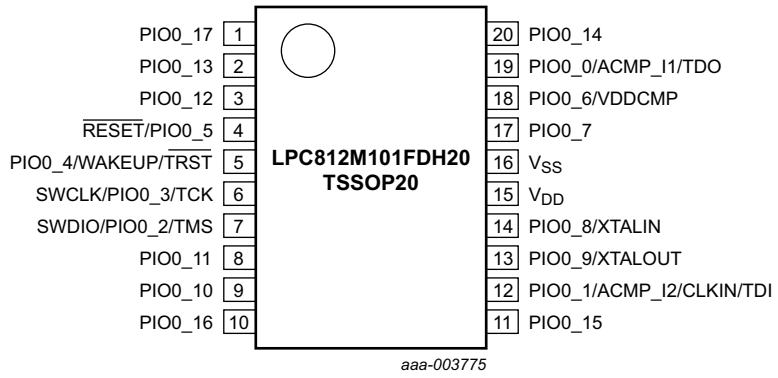


Fig 54. Pin configuration TSSOP20 package

27.2 Pin description

The pin description table Table 291 shows the pin functions that are fixed to specific pins on each package. These fixed-pin functions are selectable between the GPIO, comparator, SWD, and the XTAL pins. By default, the GPIO function is selected except on pins PIO0_2, PIO0_3, and PIO0_5. JTAG functions are available in boundary scan mode only.

Movable function for the I2C, USART, SPI, and SCT pin functions can be assigned through the switch matrix to any pin that is not power or ground in place of the pin's fixed functions.

The following exceptions apply:

For full I2C-bus compatibility, assign the I2C functions to the open-drain pins PIO0_11 and PIO0_10.

Do not assign more than one output to any pin. However, more than one input can be assigned to a pin.

Pin PIO0_4 triggers a wake-up from Deep power-down mode. If you need to wake up from Deep power-down mode via an external pin, do not assign any movable function to this pin.

The JTAG functions TDO, TDI, TCK, TMS, and $\overline{\text{TRST}}$ are selected on pins PIO0_0 to PIO0_4 by hardware when the part is in boundary scan mode.

Table 291. Pin description table (fixed pins)

Symbol	SO20/ TSSOP20	TSSOP16	DIP8		Type	Reset state [1]	Description
PIO0_0/ACMP_I1/ TDO	19	16	8	[5]	I/O	I; PU	**PIO0_0 —** General purpose digital input/output port 0 pin 0. In ISP mode, this is the USART0 receive pin U0_RXD. In boundary scan mode: TDO (Test Data Out).
					AI	-	**ACMP_I1 —** Analog comparator input 1.
PIO0_1/ACMP_I2/ CLKIN/TDI	12	9	5	[5]	I/O	I; PU	**PIO0_1 —** General purpose digital input/output pin. ISP entry pin. A LOW level on this pin during reset starts the ISP command handler. In boundary scan mode: TDI (Test Data In).
					AI	-	**ACMP_I2 —** Analog comparator input 2.
					I	-	**CLKIN —** External clock input.
SWDIO/PIO0_2/TMS	7	6	4	[2]	I/O	I; PU	**SWDIO —** Serial Wire Debug I/O. SWDIO is enabled by default on this pin. In boundary scan mode: TMS (Test Mode Select).
					I/O	-	**PIO0_2 —** General purpose digital input/output pin.
SWCLK/PIO0_3/ TCK	6	5	3	[2]	I/O	I; PU	**SWCLK —** Serial Wire Clock. SWCLK is enabled by default on this pin. In boundary scan mode: TCK (Test Clock).
					I/O	-	**PIO0_3 —** General purpose digital input/output pin.
PIO0_4/WAKEUP/ $\overline{\text{TRST}}$	5	4	2	[6]	I/O	I; PU	**PIO0_4 —** General purpose digital input/output pin. In ISP mode, this is the USART0 transmit pin U0_TXD. In boundary scan mode: $\overline{\text{TRST}}$ (Test Reset). This pin triggers a wake-up from Deep power-down mode. If you need to wake up from Deep power-down mode via an external pin, do not assign any movable function to this pin. Pull this pin HIGH externally to enter Deep power-down mode. Pull this pin LOW to exit Deep power-down mode. A LOW-going pulse as short as 50 ns wakes up the part.
$\overline{\text{RESET}}$/PIO0_5	4	3	1	[4]	I/O	I; PU	**$\overline{\text{RESET}}$ —** External reset input: A LOW-going pulse as short as 50 ns on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0.
					I	-	**PIO0_5 —** General purpose digital input/output pin.
PIO0_6/VDDCMP	18	15	-	[9]	I/O	I; PU	**PIO0_6 —** General purpose digital input/output pin.
					AI	-	**VDDCMP —** Alternate reference voltage for the analog comparator.
PIO0_7	17	14	-	[2]	I/O	I; PU	**PIO0_7 —** General purpose digital input/output pin.
PIO0_8/XTALIN	14	11	-	[8]	I/O	I; PU	**PIO0_8 —** General purpose digital input/output pin.
					I	-	**XTALIN —** Input to the oscillator circuit and internal clock generator circuits. Input voltage must not exceed 1.95 V.
PIO0_9/XTALOUT	13	10	-	[8]	I/O	I; PU	**PIO0_9 —** General purpose digital input/output pin.
					O	-	**XTALOUT —** Output from the oscillator circuit.
PIO0_10	9	8	-	[3]	I	IA	**PIO0_10 —** General purpose digital input/output pin. Assign I2C functions to this pin when true open-drain pins are needed for a signal compliant with the full I2C specification.

Table 291. Pin description table (fixed pins)

Symbol	SO20/ TSSOP20	TSSOP16	DIP8		Type	Reset state [1]	Description
PIO0_11	8	7	-	[3]	I	IA	**PIO0_11 —** General purpose digital input/output pin. Assign I2C functions to this pin when true open-drain pins are needed for a signal compliant with the full I2C specification.
PIO0_12	3	2	-	[2]	I/O	I; PU	**PIO0_12 —** General purpose digital input/output pin.
PIO0_13	2	1	-	[2]	I/O	I; PU	**PIO0_13 —** General purpose digital input/output pin.
PIO0_14	20	-	-	[7]	I/O	I; PU	**PIO0_14 —** General purpose digital input/output pin.
PIO0_15	11	-	-	[7]	I/O	I; PU	**PIO0_15 —** General purpose digital input/output pin.
PIO0_16	10	-	-	[7]	I/O	I; PU	**PIO0_16 —** General purpose digital input/output pin.
PIO0_17	1	-	-	[7]	I/O	I; PU	**PIO0_17 —** General purpose digital input/output pin.
V_{DD}	15	12	6		-	-	3.3 V supply voltage.
V_{SS}	16	13	7			-	Ground.

[1] Pin state at reset for default function: I = Input; AI = Analog Input; O = Output; PU = internal pull-up enabled (pins pulled up to full V_{DD} level); IA = inactive, no pull-up/down enabled.

[2] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors and configurable hysteresis; includes high-current output driver.

[3] True open-drain pin. I^2C-bus pins compliant with the I^2C-bus specification for I^2C standard mode, I^2C Fast-mode, and I^2C Fast-mode Plus. Do not use this pad for high-speed applications like the SPI clock.

[4] $\overline{\text{RESET}}$ 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.

[5] 5 V tolerant pin providing standard digital I/O functions with configurable modes, configurable hysteresis, and analog input. When configured as an analog input, the digital section of the pin is disabled, and the pin is not 5 V tolerant.

[6] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors and configurable hysteresis. In Deep power-down mode, pulling this pin LOW wakes up the chip.

[7] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors and configurable hysteresis.

[8] 5 V tolerant pin providing standard digital I/O functions with configurable modes, configurable hysteresis, and analog I/O for the system oscillator. When configured as an analog I/O, the digital section of the pin is disabled, and the pin is not 5 V tolerant.

[9] Not a 5 V tolerant pin due to special analog functionality. Pin provides standard digital I/O functions with configurable modes, configurable hysteresis, and analog I/O. When configured as an analog I/O, the digital section of the pin is disabled

Table 292. Movable functions (assign to pins PIO0_0 to PIO_17 through switch matrix)

Function name	Type	Description
U0_TXD	O	Transmitter output for USART0.
U0_RXD	I	Receiver input for USART0.
$\overline{\text{U0\_RTS}}$	O	Request To Send output for USART0.
$\overline{\text{U0\_CTS}}$	I	Clear To Send input for USART0.
U0_SCLK	I/O	Serial clock input/output for USART0 in synchronous mode.
U1_TXD	O	Transmitter output for USART1.
U1_RXD	I	Receiver input for USART1.
$\overline{\text{U1\_RTS}}$	O	Request To Send output for USART1.
$\overline{\text{U1\_CTS}}$	I	Clear To Send input for USART1.
U1_SCLK	I/O	Serial clock input/output for USART1 in synchronous mode.
U2_TXD	O	Transmitter output for USART2.

Table 292. Movable functions (assign to pins PIO0_0 to PIO_17 through switch matrix)

Function name	Type	Description
U2_RXD	I	Receiver input for USART2.
$\overline{\text{U2\_RTS}}$	O	Request To Send output for USART2.
$\overline{\text{U2\_CTS}}$	I	Clear To Send input for USART2.
U2_SCLK	I/O	Serial clock input/output for USART2 in synchronous mode.
SPI0_SCK	I/O	Serial clock for SPI0.
SPI0_MOSI	I/O	Master Out Slave In for SPI0.
SPI0_MISO	I/O	Master In Slave Out for SPI0.
SPI0_SSEL	I/O	Slave select for SPI0.
SPI1_SCK	I/O	Serial clock for SPI1.
SPI1_MOSI	I/O	Master Out Slave In for SPI1.
SPI1_MISO	I/O	Master In Slave Out for SPI1.
SPI1_SSEL	I/O	Slave select for SPI1.
CTIN_0	I	SCT input 0.
CTIN_1	I	SCT input 1.
CTIN_2	I	SCT input 2.
CTIN_3	I	SCT input 3.
CTOUT_0	O	SCT output 0.
CTOUT_1	O	SCT output 1.
CTOUT_2	O	SCT output 2.
CTOUT_3	O	SCT output 3.
I2C0_SCL	I/O	I^2C-bus clock input/output (open-drain if assigned to pin PIO0_10). High-current sink only if assigned to PIO0_10 and if I^2C Fast-mode Plus is selected in the I/O configuration register.
I2C0_SDA	I/O	I^2C-bus data input/output (open-drain if assigned to pin PIO0_11). High-current sink only if assigned to pin PIO0_11 and if I^2C Fast-mode Plus is selected in the I/O configuration register.
ACMP_O	O	Analog comparator output.
CLKOUT	O	Clock output.
GPIO_INT_BMAT	O	Output of the pattern match engine.

28.1 How to read this chapter

This chapter summarizes the ARM Cortex-M0+ instructions. The instruction set is identical for all LPC800 parts.

28.2 General description

The processor implements the ARMv6-M Thumb instruction set, including a number of 32-bit instructions that use Thumb-2 technology. The ARMv6-M instruction set contains:

- all of the 16-bit Thumb instructions from ARMv7-M excluding CBZ, CBNZ and IT.
- the 32-bit Thumb instructions BL, DMB, DSB, ISB, MRS and MSR.

Table 293 shows the Cortex-M0+ instructions and their cycle counts. The cycle counts are based on a system with zero wait-states.

Table 293. Cortex M0- instruction summary

Operation	Description	Assembler	Cycles
Move	8-bit immediate	MOVS Rd, #<imm>	1
	Lo to Lo	MOVS Rd, Rm	1
	Any to Any	MOV Rd, Rm	1
	Any to PC	MOV PC, Rm	2
Add	3-bit immediate	ADDS Rd, Rn, #<imm>	1
	All registers Lo	ADDS Rd, Rn, Rm	1
	Any to Any	ADD Rd, Rd, Rm	1
	Any to PC	ADD PC, PC, Rm	2
	8-bit immediate	ADDS Rd, Rd, #<imm>	1
	With carry	ADCS Rd, Rd, Rm	1
	Immediate to SP	ADD SP, SP, #<imm>	1
	Form address from SP	ADD Rd, SP, #<imm>	1
	Form address from PC	ADR Rd, <label>	1
Subtract	Lo and Lo	SUBS Rd, Rn, Rm	1
	3-bit immediate	SUBS Rd, Rn, #<imm>	1
	8-bit immediate	SUBS Rd, Rd, #<imm>	1
	With carry	SBCS Rd, Rd, Rm	1
	Immediate from SP	SUB SP, SP, #<imm>	1
	Negate	RSBS Rd, Rn, #0	1
Multiply	Multiply	MULS Rd, Rm, Rd	1
Compare	Compare	CMP Rn, Rm	1
	Negative	CMN Rn, Rm	1
	Immediate	CMP Rn, #<imm>	1

Table 293. Cortex M0- instruction summary

Operation	Description	Assembler	Cycles
Logical	AND	ANDS Rd, Rd, Rm	1
	Exclusive OR	EORS Rd, Rd, Rm	1
	OR	ORRS Rd, Rd, Rm	1
	Bit clear	BICS Rd, Rd, Rm	1
	Move NOT	MVNS Rd, Rm	1
	AND test	TST Rn, Rm	1
Shift	Logical shift left by immediate	LSLS Rd, Rm, #<shift>	1
	Logical shift left by register	LSLS Rd, Rd, Rs	1
	Logical shift right by immediate	LSRS Rd, Rm, #<shift>	1
	Logical shift right by register	LSRS Rd, Rd, Rs	1
	Arithmetic shift right	ASRS Rd, Rm, #<shift>	1
	Arithmetic shift right by register	ASRS Rd, Rd, Rs	1
Rotate	Rotate right by register	RORS Rd, Rd, Rs	1
Load	Word, immediate offset	LDR Rd, [Rn, #<imm>]	2 or 1[2]
	Halfword, immediate offset	LDRH Rd, [Rn, #<imm>]	2 or 1[2]
	Byte, immediate offset	LDRB Rd, [Rn, #<imm>]	2 or 1[2]
	Word, register offset	LDR Rd, [Rn, Rm]	2 or 1[2]
	Halfword, register offset	LDRH Rd, [Rn, Rm]	2 or 1[2]
	Signed halfword, register offset	LDRSH Rd, [Rn, Rm]	2 or 1[2]
	Byte, register offset	LDRB Rd, [Rn, Rm]	2 or 1[2]
	Signed byte, register offset	LDRSB Rd, [Rn, Rm]	2 or 1[2]
	PC-relative	LDR Rd, <label>	2 or 1[2]
	SP-relative	LDR Rd, [SP, #<imm>]	2 or 1[2]
	Multiple, excluding base	LDM Rn!, {<loreglist>}	1 + N[1]
	Multiple, including base	LDM Rn, {<loreglist>}	1 + N[1]
Store	Word, immediate offset	STR Rd, [Rn, #<imm>]	2 or 1[2]
	Halfword, immediate offset	STRH Rd, [Rn, #<imm>]	2 or 1[2]
	Byte, immediate offset	STRB Rd, [Rn, #<imm>]	2 or 1[2]
	Word, register offset	STR Rd, [Rn, Rm]	2 or 1[2]
	Halfword, register offset	STRH Rd, [Rn, Rm]	2 or 1[2]
	Byte, register offset	STRB Rd, [Rn, Rm]	2 or 1[2]
	SP-relative	STR Rd, [SP, #<imm>]	2 or 1[2]
	Multiple	STM Rn!, {<loreglist>}	1 + N[1]
Push	Push	PUSH {<loreglist>}	1 + N[1]
	Push with link register	PUSH {<loreglist>, LR}	1 + N[3]
Pop	Pop	POP {<loreglist>}	1 + N[1]
	Pop and return	POP {<loreglist>, PC}	3 + N[3]

Table 293. Cortex M0- instruction summary

Operation	Description	Assembler	Cycles
Branch	Conditional	B<cc> <label>	1 or 2[4]
	Unconditional	B <label>	2
	With link	BL <label>	3
	With exchange	BX Rm	2
	With link and exchange	BLX Rm	2
Extend	Signed halfword to word	SXTH Rd, Rm	1
	Signed byte to word	SXTB Rd, Rm	1
	Unsigned halfword	UXTH Rd, Rm	1
	Unsigned byte	UXTB Rd, Rm	1
Reverse	Bytes in word	REV Rd, Rm	1
	Bytes in both halfwords	REV16 Rd, Rm	1
	Signed bottom half word	REVSH Rd, Rm	1
State change	Supervisor Call	SVC #<imm>	-[5]
	Disable interrupts	CPSID i	1
	Enable interrupts	CPSIE i	1
	Read special register	MRS Rd, <specreg>	3
	Write special register	MSR <specreg>, Rn	3
	Breakpoint	BKPT #<imm>	-[5]
Hint	Send event	SEV	1
	Wait for interrupt	WFI	2[6]
	Wait for event	WFE	2[6]
	Yield	YIELD[7]	1
	No operation	NOP	1
Barriers	Instruction synchronization	ISB	3
	Data memory	DMB	3
	Data synchronization	DSB	3

[1] N is the number of elements in the list.

[2] 2 cycles if to AHB interface or SCS, 1 cycle if to single-cycle I/O port.

[3] N is the number of elements in the list including PC or LR..

[4] 2 if taken, 1 if not taken.

[5] Cycle count depends on core and debug configuration.

[6] Excludes time spend waiting for an interrupt or event.

[7] Executes as NOP.

29.1 Abbreviations

Table 294. Abbreviations

Acronym	Description
AHB	Advanced High-performance Bus
APB	Advanced Peripheral Bus
BOD	BrownOut Detection
GPIO	General-Purpose Input/Output
PLL	Phase-Locked Loop
RC	Resistor-Capacitor
SPI	Serial Peripheral Interface
SMBus	System Management Bus
TEM	Transverse ElectroMagnetic
UART	Universal Asynchronous Receiver/Transmitter

29.2 References

[1] **DDI0484B_cortex_m0p_r0p0_trm —** ARM Cortex-M0+ Technical Reference Manual

[2] **DDI0486A —** ARM technical reference manual

[3] ARMv6-M Architecture Reference Manual

29.3 Legal information

29.3.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

29.3.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

29.3.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

I^2C-bus — logo is a trademark of NXP B.V.

29.4 Tables

29.5 Figures

29.6 Contents

Chapter 6: LPC800 I/O configuration (IOCON)

Chapter 7: LPC800 GPIO port

Chapter 8: LPC800 Pin interrupts/pattern match engine

Chapter 9: LPC800 Switch matrix

Chapter 10: LPC800 State Configurable Timer (SCT)

Chapter 11: LPC800 Multi-Rate Timer (MRT)

Chapter 12: LPC800 Windowed Watchdog Timer (WWDT)

Chapter 13: LPC800 Self wake-up timer (WKT)

Chapter 14: LPC800 ARM Cortex SysTick Timer (SysTick)

Chapter 15: LPC800 USART0/1/2

Chapter 16: LPC800 I2C-bus interface

Chapter 17: LPC800 SPI0/1

Chapter 18: LPC800 Analog comparator

Chapter 19: LPC800 Cyclic Redundancy Check (CRC) engine

Chapter 20: LPC800 Flash controller

Chapter 21: LPC800 Boot ROM

Chapter 22: LPC800 Flash ISP and IAP programming

Chapter 23: LPC800 Power profile API ROM driver

Chapter 24: LPC800 I2C-bus ROM API

Chapter 25: LPC800 USART API ROM driver routines

Chapter 26: LPC800 Debugging

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 14 March 2013
Document identifier: UM10601