UM10732

LPC11U6x/E6x User manual

Rev. 1.8 — 19 December 2016 User manual

Document information

Info	Content
Keywords	LPC11U6x user manual, LPC11U6x UM, LPC11U67, LPC11U68, USB, ARM Cortex M0+, LPC11E6x, LPC11E67, LPC11E68, LPC11U66, LPC11E66
Abstract	LPC11U6x/E6x User manual

Revision history

Rev	Date	Description
1.8	20161219	
		• Added part number LPC11E66 to Section 12.1 “How to read this chapter”.
1.7	20161012	LPC11U6x/E6x user manual
		• Added text after Table 246 “Endpoint commands”: For EP0 transfers, the hardware will do auto handshake as long as the ACTIVE bit is set in EP0_IN/OUT command list. Unlike other endpoints, the hardware will not clear the ACTIVE bit after transfer is done. Thus, the software should manually clear the bit whenever it receives new setup packet and set it only after it has queued the data for control transfer.
1.6	20160906	LPC11U6x/E6x user manual
		• Updated Figure 3 “AHB multilayer matrix”: HS GPIO connects with M0+ Core and not with DMA or USB.
1.5	20160825	LPC11U6x/E6x user manual
		• Replaced #define IAP_LOCATION 0x1FFF1FF8 with #define IAP_LOCATION 0x1FFF1FF1. See Section 27.6 “API description (IAP)”. • Replaced the text for flash signature generator as 128-bits. See Section 25.5.1 “Flash signature generation” • Changed the address of the device id register to 0x400483F8. See Table 70 “Device ID (DEVICE_ID, address 0x400483F8) bit description”. • Changed the text: This register enables the 32 kHz (31.768 kHz) output of the RTC oscillator to This register enables the 32 kHz (32.768 kHz) output of the RTC oscillator. See Section 4.4.7 “RTC oscillator 32 kHz output control register”. • Changed the title DMA trigger input mux registers 0 to 17 to DMA trigger input mux registers 0 to 15. See Section 10.5.19. • Updated Table 148: – Bit field INP_N covers bits 3:0; was 4:0. – Reserved bits are 31:4. – Reset value is 0xF; was 0x1F. – Trigger input 5 is CT32B1_MAT0; was CT16B1_MAT0.

Revision history *…continued*

Rev	Date	Description
1.4	20151028	LPC11U6x/E6x user manual

- Updated autobaud description (clearance of AUTOBAUD bit). See Section 12.7.5 “Autobaud function”.
- For autobaud, set BRG to 0x0 (default value). See Section 12.7.5 “Autobaud function”.
- VREFP/VREFN voltage conditions added in Section 16.4.
- Updated the flash access times in Table 354 “Flash configuration register (FLASHCFG, address 0x4003 C010) bit description”.
- Updated cross reference to FIFO control register (FCR) to point to the correct table. See Table 150 “Register overview: USART0 (base address: 0x4000 8000)”
- Fixed the polarity for REGMODE in the SCTimer/PWM chapter. REGMODEn = 1: Registers operate as capture and capture control registers. REGMODEn = 0: Registers operate as match and reload registers. See List item 2 “The REGMODEn bits in the REGMODE register determine whether each set of Match/Capture registers uses the match or capture functionality:”
- Replaced CT16B0_CAP1 with CT16B0_CAP2 for pin PIO1_21 in Table 81 “Register overview: I/O configuration (base address 0x4004 4000)”, Table 83 “IOCON function assignments”Table 494 “Pin description”, and Table 495 “Pin description”.
- Replaced CT32B0_CAP1 with CT32B0_CAP2 for pin PIO1_6 and pin PIO1_29 in Table 81 “Register overview: I/O configuration (base address 0x4004 4000)”, Table 83 “IOCON function assignments”Table 494 “Pin description”, and Table 495 “Pin description”.
- Added a note to the descriptions of the SCT DMA flags that they will likely not be seen by software if the DMA is correctly configured and enabled. see Table 281 “SCT DMA 0 request register (DMAREQ0, address 0x5000 C05C (SCT0) and 0x5000 E05C (SCT1)) bit description” and Table 282 “SCT DMA 1 request register (DMAREQ1, address 0x5000 C060 (SCT0) and 0x5000 E060 (SCT1)) bit description”.

Revision history *...continued*

Rev	Date	Description
1.3	20140519	LPC11U6x/E6x user manual
Modifications		- Updated ADC calibration routine: Calibration is only required after wake-up from deep-power down mode and after power-up. - Parts added: LPC11U66JBD48, LPC11U67JBD64, LPC11U67JBD100, LPC11E66JBD48, LPC11E67JBD64, LPC11E67JBD100, LPC11E68JBD48. - Bit description of the AUTOBAUD bit updated in Table 181 “USART Control register (CTL, address 0x4006 C004 (USART1), 0x4007 0004 (USART2), 0x4007 4004 (USART3), 0x4004 C004 (USART4)) bit description”: This bit can only be set when the UART is enabled in the CFG register and is cleared when the UART is disabled. - RTC oscillator frequencies described accurately: 32.768 kHz and 1.024 kHz for 32 kHz and 1 kHz modes. See Chapter 21.
1.2	20140404	LPC11U6x/E6x user manual
		- Part ID added for part LPC11U68JBD48. - Figure 86 “Boot process flowchart” corrected. - Watchdog interrupt flag polarity corrected: This flag is cleared by writing a 1 to the WDINT bit in the MOD register (Section 22.5.1 “Watchdog mode register”). - Use of IAP mode with power profiles clarified. Use power profiles in default mode when executing IAP commands. See Section 27.6 “API description (IAP)” and Section 28.3. - Section 28.3 added to clarify use of power profiles. - Table 31 “Internal resonant crystal control register (IRCCTRL, address 0x4004 8028) bit description” added.
1.1	20140304	LPC11U6x/E6x user manual
Modifications:		- Size of parameter driver_mode changed to uint8_t in UART_PARAM_T structure. See Section 31.4.10.3 and Section 32.4.10.3. - Table 295 “SCT configuration example” corrected. - IOCON function bits corrected for registers TDI_PIO0_11, TMS_PIO0_12, TDO_PIO0_13, TRST_PIO0_14. See Table 82 “IOCON function assignments”. - Description of SCT HALT bit behavior in dual-counter mode added. See Table 268 “SCT control register (CTRL, address 0x5000 C004 (SCT0) and 0x5000 E004 (SCT1)) bit description”, Table 292, and Table 293. - Section 26.5.3 “Boot process” and Figure 86 “Boot process flowchart” corrected: The part enumerates as USB MSC device when no valid user code is present in flash. - Section 11.7.6 “USART clock in synchronous mode” added. - Remark added to Section 5.5.5.3 “Wake-up from Deep-sleep mode” and Section 5.5.6.3 “Wake-up from Power-down mode”: **Remark:** After wake-up, reprogram the clock source for the main clock.
1	20140114	Initial LPC11U6x/E6x user manual version.

Contact information

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

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

UM10732

Chapter 1: LPC11U6x/E6x Introductory information

Rev. 1.8 — 19 December 2016 **User manual**

1.1 Introduction

The LPC11U6x/Ex are an ARM Cortex-M0+ based, low-cost 32-bit MCU family operating at CPU frequencies of up to 50 MHz. The LPC11U6x/Ex support up to 256 KB of flash memory, a 4 KB EEPROM, and up to 36 KB of SRAM.

The ARM Cortex-M0+ is an easy-to-use, energy-efficient core using a two-stage pipeline and fast single-cycle I/O access.

The peripheral complement of the LPC11U6x/Ex includes a DMA controller, a CRC engine, one full-speed USB device controller with XTAL-less low-speed mode (LPC11U6x only), two I^{2}C-bus interfaces, up to five USARTs, two SSP interfaces, PWM/timer subsystem with six configurable multi-purpose timers, a Real-Time Clock, one 12-bit ADC, temperature sensor, function-configurable I/O ports, and up to 80 general-purpose I/O pins.

For additional documentation related to the LPC11U6x/E6x parts, see Section 37.2 “References”.

1.2 Features

- System:
    - ARM Cortex-M0+ processor, running at frequencies of up to 50 MHz with single-cycle multiplier and fast single-cycle I/O port.
    - ARM Cortex-M0+ built-in Nested Vectored Interrupt Controller (NVIC).
    - AHB Multilayer matrix.
    - System tick timer.
    - Serial Wire Debug (SWD) and JTAG boundary scan modes supported.
    - Micro Trace Buffer (MTB) supported.
- Memory:
    - Up to 256 KB on-chip flash programming memory with page erase.
    - Up to 32 KB main SRAM.
    - Up to two additional SRAM blocks of 2 KB each.
    - Up to 4 KB EEPROM.
- ROM API support:
    - Boot loader.
    - USART drivers.
    - I2C drivers.
    - USB drivers (LPC11U6x only).
    - DMA drivers.
    - Power profiles.
    - Flash In-Application Programming (IAP) and In-System Programming (ISP).

  - 32-bit integer division routines.
- Digital peripherals:
  - Simple DMA engine with 16 channels and programmable input triggers.
  - High-speed GPIO interface connected to the ARM Cortex-M0+ IO bus with up to 80 General-Purpose I/O (GPIO) pins with configurable pull-up/pull-down resistors, programmable open-drain mode, input inverter, and programmable glitch filter and digital filter.
  - Pin interrupt and pattern match engine using eight selectable GPIO pins.
  - Two GPIO group interrupt generators.
  - CRC engine.
- Configurable PWM/timer subsystem (two 16-bit and two 32-bit standard counter/timers, two State-Configurable Timers (SCTimer/PWM)) that provides:
  - Up to four 32-bit and two 16-bit counter/timers or two 32-bit and six 16-bit counter/timers.
  - Up to 21 match outputs and 16 capture inputs.
  - Up to 19 PWM outputs with 6 independent time bases.
- Windowed Watchdog timer (WWDT).
- Real-time Clock (RTC) in the always-on power domain with separate battery supply pin and 32.768 kHz oscillator.
- Analog peripherals:
  - One 12-bit ADC with up to 12 input channels with multiple internal and external trigger inputs and with sample rates of up to 2 Msamples/s. The ADC supports two independent conversion sequences.
  - Temperature sensor.
- Serial interfaces:
  - Up to five USART interfaces, all with DMA, synchronous mode, and RS-485 mode support. Four USARTs use a shared fractional baud generator.
  - Two SSP controllers with DMA support.
  - Two I^{2}C-bus interfaces. One I^{2}C-bus interface with specialized open-drain pins supports I2C Fast-mode plus.
  - USB 2.0 full-speed device controller with on-chip PHY. XTAL-less low-speed mode supported (LPC11U6x only).
- Clock generation:
  - 12 MHz internal RC oscillator trimmed to 1 % accuracy for $-25\ °C \leq T_{amb} \leq +85\ °C$ that can optionally be used as a system clock.
  - On-chip 32.768 kHz oscillator for RTC.
  - Crystal oscillator with an operating range of 1 MHz to 25 MHz. Oscillator pins are shared with the GPIO pins.
  - Programmable watchdog oscillator with a frequency range of 9.4 kHz to 2.3 MHz.
  - PLL allows CPU operation up to the maximum CPU rate without the need for a high-frequency crystal.
  - A second, dedicated PLL is provided for USB (LPC11U6x only).

  - 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 external pin inputs and USART activity.
  - Power-On Reset (POR).
  - Brownout detect.
- Unique device serial number for identification.
- Single power supply (2.4 V to 3.6 V).
- Separate VBAT supply for RTC.
- Operating temperature range -40 °C to +105 °C.
- Available as LQFP48, LQFP64, and LQFP100 packages.

1.3 Ordering information

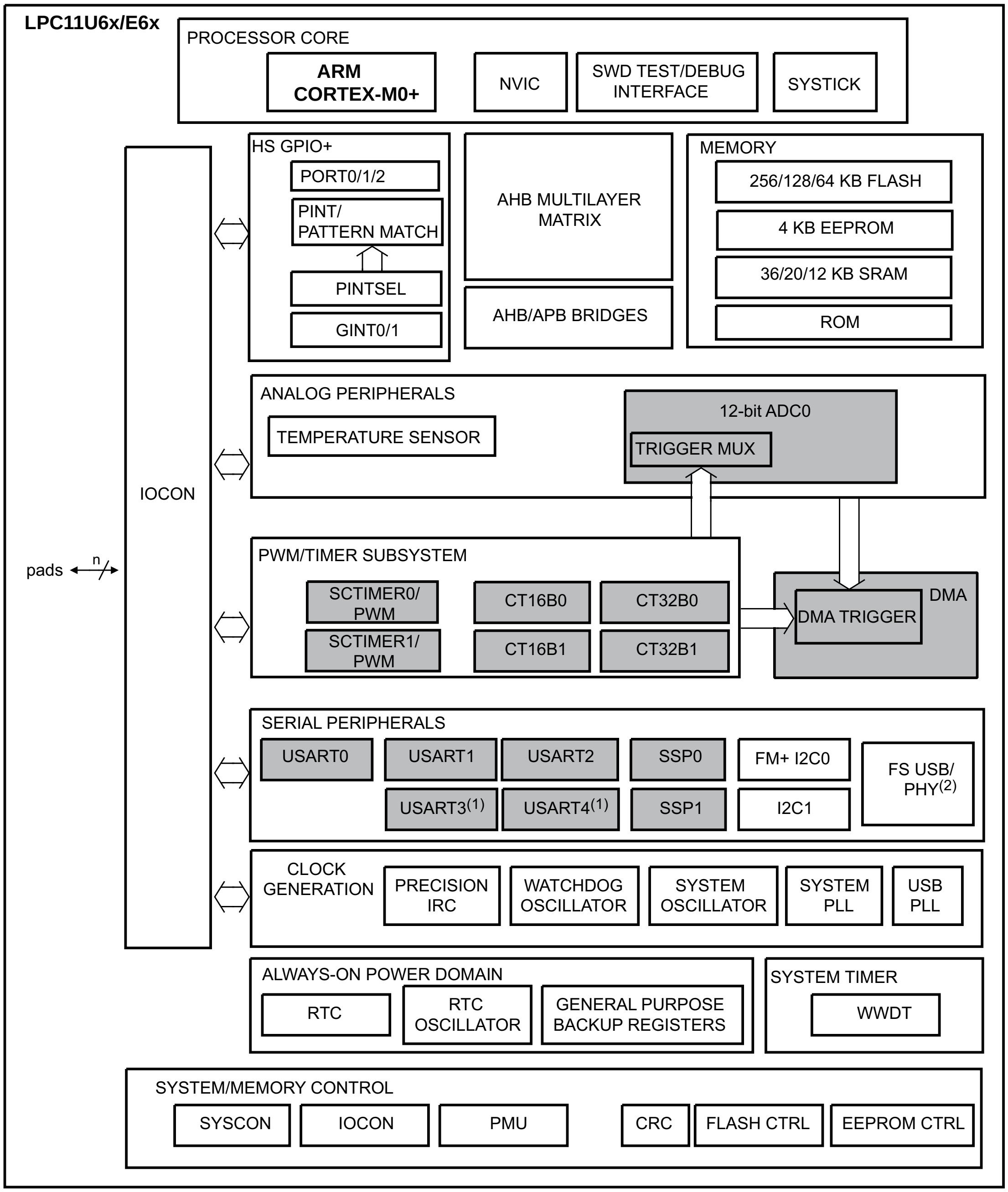
Table 1. Ordering information

Type number	Package		
	Name	Description	Version
LPC11U66JBD48	LQFP48	plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2
LPC11U67JBD48	LQFP48	plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2
LPC11U67JBD64	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC11U67JBD100	LQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 1.4 mm	SOT407-1
LPC11U68JBD48	LQFP48	plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2
LPC11U68JBD64	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC11U68JBD100	LQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 1.4 mm	SOT407-1
LPC11E66JBD48	LQFP48	plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2
LPC11E67JBD48	LQFP48	plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2
LPC11E67JBD64	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC11E67JBD100	LQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 1.4 mm	SOT407-1
LPC11E68JBD48	LQFP48	plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2
LPC11E68JBD64	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC11E68JBD100	LQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 1.4 mm	SOT407-1

Table 2. Ordering options

Type number	Flash /KB	EEPROM/ KB	SRAM/ KB	USB	USART0	USART1	USART2	USART3	USART4	I^{2}C	SSP	Timers with PWM	12-bit ADC channels	GPIO
LPC11U66JBD48	64	4	12	1	Y	Y	Y	N	N	2	2	6	8	34
LPC11U67JBD48	128	4	20	1	Y	Y	Y	N	N	2	2	6	8	34
LPC11U67JBD64	128	4	20	1	Y	Y	Y	N	N	2	2	6	10	48
LPC11U67JBD100	128	4	20	1	Y	Y	Y	Y	Y	2	2	6	12	80
LPC11U68JBD48	256	4	36	1	Y	Y	Y	N	N	2	2	6	8	34
LPC11U68JBD64	256	4	36	1	Y	Y	Y	N	N	2	2	6	10	48
LPC11U68JBD100	256	4	36	1	Y	Y	Y	Y	Y	2	2	6	12	80
LPC11E66JBD48	64	4	12	-	Y	Y	Y	Y	N	2	2	6	8	36
LPC11E67JBD48	128	4	20	-	Y	Y	Y	Y	N	2	2	6	8	36
LPC11E67JBD64	128	4	20	-	Y	Y	Y	Y	N	2	2	6	10	50
LPC11E67JBD100	128	4	20	-	Y	Y	Y	Y	Y	2	2	6	12	80
LPC11E68JBD48	256	4	36	-	Y	Y	Y	Y	N	2	2	6	8	36
LPC11E68JBD64	256	4	36	-	Y	Y	Y	Y	N	2	2	6	10	50
LPC11E68JBD100	256	4	36	-	Y	Y	Y	Y	Y	2	2	6	12	80

1.4 Block diagram



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

(1) Available on LPC11U68JBD100/E68JBD100 only.

(2) Available on LPC11U6x only.

Fig 1. LPC11U6x 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 50 MHz. Integrated in the core are the NVIC and Serial Wire Debug with four breakpoints and two watch points. The ARM Cortex-M0+ core supports a single-cycle I/O enabled port (IOP) for fast GPIO access at address 0xA000 0000. The ARM Cortex M0+ core revision is r0p1.

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

1.5.2 Timer/PWM subsystem

Four standard timers and two state configurable timers can be combined to create multiple PWM outputs using the match outputs and the match registers for each timers. Each timer can create multiple PWM outputs with its own time base.

Table 3. PWM resources

PWM outputs			Peripheral	Pin functions available for PWM			Match registers used
LQFP100	LQFP64	LQFP48		LQFP100	LQFP64	LQFP48	
3	3	3	CT16B0	CT16B0_MAT0, CT16B0_MAT1, CT16B0_MAT2	CT16B0_MAT0, CT16B0_MAT1, CT16B0_MAT2	CT16B0_MAT0, CT16B0_MAT1, CT16B0_MAT2	4
2	2	2	CT16B1	CT16B1_MAT0, CT16B1_MAT1	CT16B1_MAT0, CT16B1_MAT1	CT16B1_MAT0, CT16B1_MAT1	3
3	3	3	CT32B0	three of CT32B0_MAT0, CT32B0_MAT1, CT32B0_MAT2, CT32B0_MAT3	three of CT32B0_MAT0, CT32B0_MAT1, CT32B0_MAT2, CT32B0_MAT3	three of CT32B0_MAT0, CT32B0_MAT1, CT32B0_MAT2, CT32B0_MAT3	4
3	3	3	CT32B1	three of CT32B1_MAT0, CT32B1_MAT1, CT32B1_MAT2, CT32B1_MAT3	three of CT32B1_MAT0, CT32B1_MAT1, CT32B1_MAT2, CT32B1_MAT3	three of CT32B1_MAT0, CT32B1_MAT1, CT32B1_MAT2, CT32B1_MAT3	4
4	4	3	SCTimer0/ PWM	SCT0_OUT0, SCT0_OUT1, SCT0_OUT2, SCT0_OUT3	SCT0_OUT0, SCT0_OUT1, SCT0_OUT2, SCT0_OUT3	SCT0_OUT1, SCT0_OUT2, SCT0_OUT3	up to 5
4	2	-	SCTimer1/ PWM	SCT1_OUT0, SCT1_OUT1, SCT1_OUT2, SCT1_OUT3	SCT1_OUT2, SCT1_OUT3	-	up to 5

The standard timers and the SCTimers combine to up to eight independent timers. Each STimer can be configured either as one 32-bit timer or two independently counting 16-bit timers which use the same input clock. The following combinations are possible:

Table 4. Timer configurations

32-bit timers	Resources	16-bit timers	Resources
4	CT32B0, CT32B1, SCTimer0 as 32-bit timer, SCTimer1 as 32-bit timer	2	CT16B0, CT16B1
2	CT32B0, CT32B1	6	CT16B0, CT16B1, SCTimer0 as two 16-bit timers, SCTimer1 as two 16-bit timers
3	CT32B0, CT32B1, SCTimer0 as 32-bit timer (or SCTimer1 as 32-bit timer)	4	CT16B0, CT16B1, SCTimer1 as two 16-bit timers (or SCTimer0 as two 16-bit timers)

UM10732

Chapter 2: LPC11U6x/E6x Memory map

Rev. 1.8 — 19 December 2016 **User manual**

2.1 How to read this chapter

See Table 5 for the memory configuration of the LPC11U6x/E6x parts. The USB interface is only available on LPC11U6x parts.

Table 5. Memory configuration

Type number	Flash/ KB	SRAM/KB			EEPROM/KB
		Main SRAM0 at 0x1000 0000	SRAM1 at 0x2000 0000	USB SRAM/SRAM2 at 0x2000 4000	
LPC11U66JBD48	64	8	2	2	4
LPC11U67JBD48	128	16	2	2	4
LPC11U67JBD64	128	16	2	2	4
LPC11U67JBD100	128	16	2	2	4
LPC11U68JBD48	256	32	2	2	4
LPC11U68JBD64	256	32	2	2	4
LPC11U68JBD100	256	32	2	2	4
LPC11E66JBD48	64	8	2	2	4
LPC11E67JBD48	128	16	2	2	4
LPC11E67JBD64	128	16	2	2	4
LPC11E67JBD100	128	16	2	2	4
LPC11E68JBD48	256	32	2	2	4
LPC11E68JBD64	256	32	2	2	4
LPC11E68JBD100	256	32	2	2	4

2.2 Basic configuration

The SRAM0 block, the USB SRAM/SRAM2 block, flash memory, and EEPROM are enabled by default. The user code must enable the clock to the SRAM1 block in the SYSAHBCLKCTRL register.

2.3 General description

The part incorporates several distinct memory regions, shown in the following figures. 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).

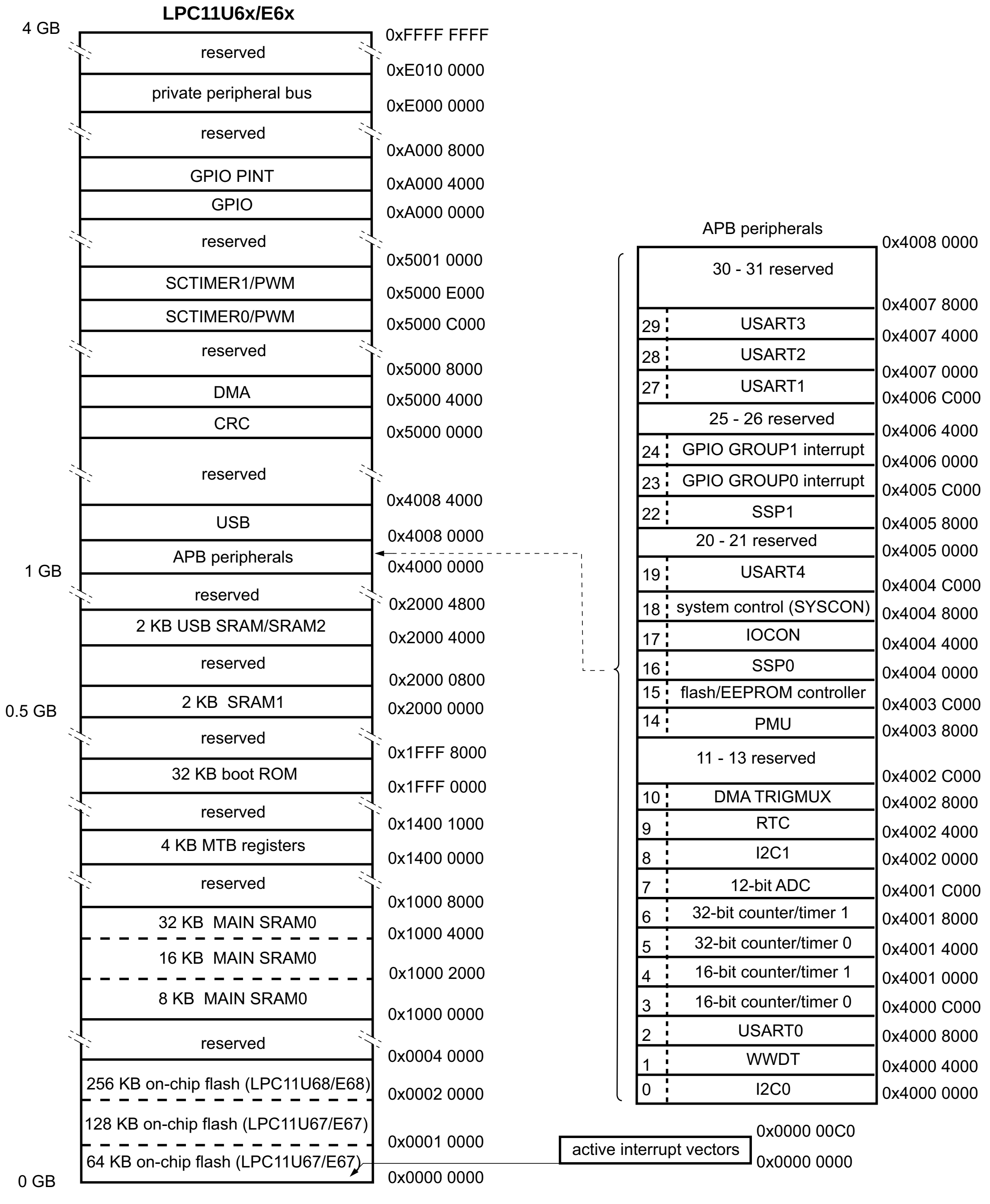


Fig 2. **LPC11U6x/E6x memory mapping**

2.3.1 On-chip flash programming memory

The part contains up to 256 KB on-chip flash program memory. The flash can be programmed using In-System Programming (ISP) or In-Application Programming (IAP) via the on-chip boot loader software. Flash updates via USB are supported as well.

The flash memory is divided into 24 x 4 KB and 5 x 32 KB sectors. Individual pages of 256 byte each can be erased using the IAP erase page command.

2.3.2 EEPROM

The LPC11U6x/E6x contain up to 4 KB of on-chip byte-erasable and byte-programmable EEPROM data memory. The EEPROM can be programmed using In-Application Programming (IAP) via the on-chip boot loader software.

2.3.3 SRAM

The LPC11U6x/E6x contain a total of up to 36 KB of on-chip static RAM memory. See Table 5 for the memory configuration for each part.

The SRAM1 clock is turned off by default. Enable the clock in the SYSAHBCLKCTRL register (Table 40).

2.3.4 Micro Trace Buffer (MTB)

The LPC11U6x/E6x supports the ARM Cortex-M0+ Micro Trace Buffer. See Section 36.5.4.

2.3.5 AHB multilayer matrix

The AHB multilayer matrix supports three masters, the M0+ core, the DMA, and the USB. All masters can access all slaves (peripherals and memories).

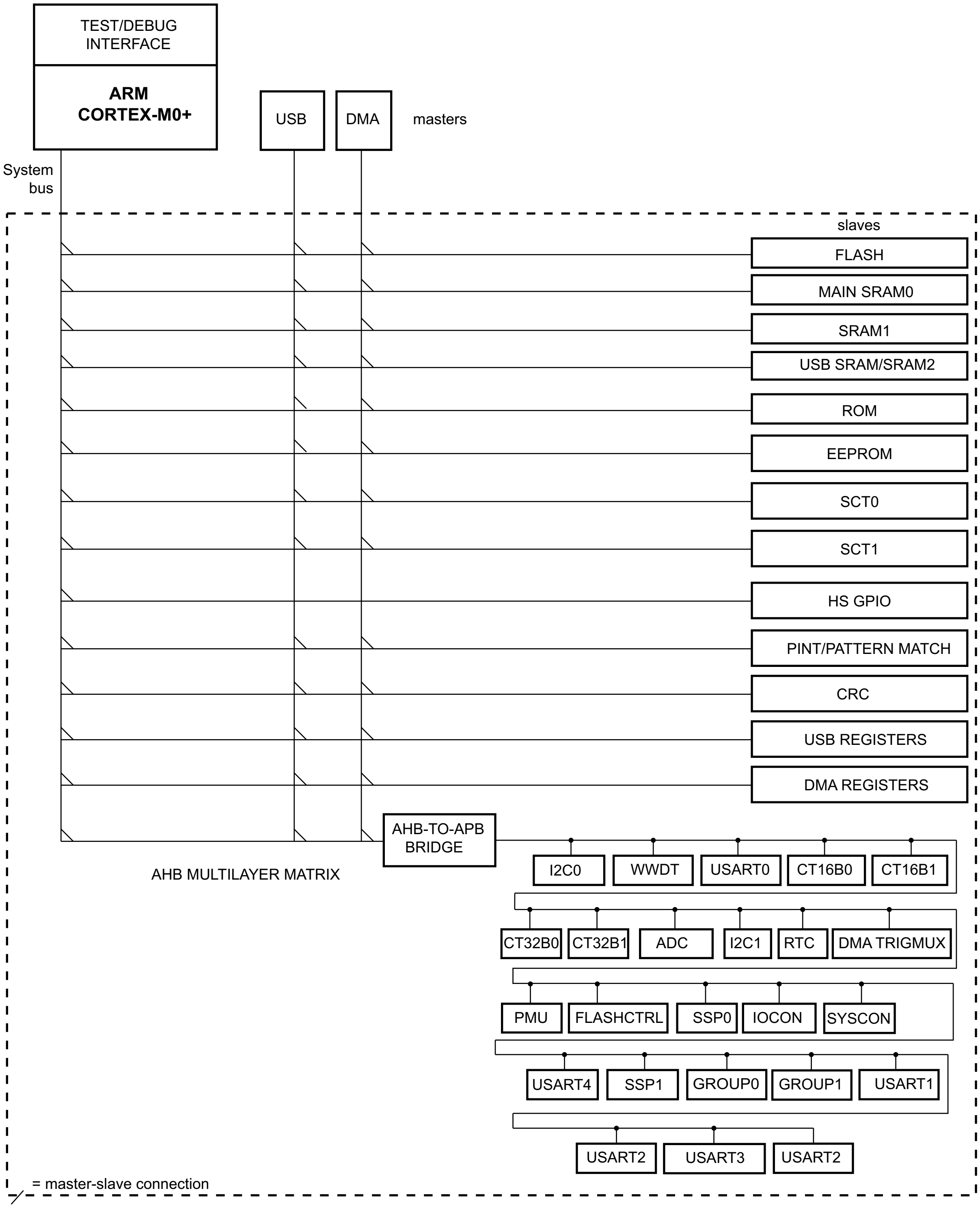


Fig 3. AHB multilayer matrix

UM10732

Chapter 3: LPC11U6x/E6x Nested Vectored Interrupt Controller (NVIC)

Rev. 1.8 — 19 December 2016 **User manual**

3.1 How to read this chapter

The NVIC is identical on all parts. The USB interrupts are available on LPC11U6x only.

3.2 Features

- Nested Vectored Interrupt Controller 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.
- Support for NMI.

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 External pin interrupts

Up to eight external pin interrupts are supported. Each of the eight pin interrupts can be assigned to any pin on port 0 (PIO0), any pin on port 1 (PIO1), or pins 0 to 7 on port 2 (PIO2_0 to PIO2_7). See Table 62.

3.3.2 Interrupt sources

Table 6 lists the interrupt sources for each peripheral function. Each peripheral device may have one or more interrupt lines to the Vectored Interrupt Controller. Each line may represent more than one interrupt source. There is no significance or priority about what line is connected where, except for certain standards from ARM.

Table 6. Connection of interrupt sources to the Vectored Interrupt Controller

Interrupt number	Name	Description	Flag(s)
0	PIN_INT0	GPIO pin interrupt 0	PSTAT - pin interrupt status
1	PIN_INT1	GPIO pin interrupt 1	PSTAT - pin interrupt status
2	PIN_INT2	GPIO pin interrupt 2	PSTAT - pin interrupt status
3	PIN_INT3	GPIO pin interrupt 3	PSTAT - pin interrupt status
4	PIN_INT4	GPIO pin interrupt 4	PSTAT - pin interrupt status
5	PIN_INT5	GPIO pin interrupt 5	PSTAT - pin interrupt status
6	PIN_INT6	GPIO pin interrupt 6	PSTAT - pin interrupt status
7	PIN_INT7	GPIO pin interrupt 7	PSTAT - pin interrupt status
8	GINT0	GPIO GROUP0 interrupt	INT - group interrupt status
9	GINT1	GPIO GROUP1 interrupt	INT - group interrupt status
10	I2C1	I2C1 interrupt	SI (state change)
11	USART1_4	Combined USART1 and USART4 interrupts	Table 189 “USART Interrupt Status register (INTSTAT, address 0x4006 C024 (USART1), 0x4007 0024 (USART2), 0x4007 4024 (USART3), 0x4004 C024 (USART4)) bit description”
12	USART2_3	Combined USART2 and USART3 interrupts	Table 189 “USART Interrupt Status register (INTSTAT, address 0x4006 C024 (USART1), 0x4007 0024 (USART2), 0x4007 4024 (USART3), 0x4004 C024 (USART4)) bit description”
13	SCT0_1	Combined SCT0 and SCT1 interrupts	EVFLAG SCT event.
14	SSP1	SSP1 interrupt	Tx FIFO half empty Rx FIFO half full Rx Timeout Rx Overrun
15	I2C0	I2C0 interrupt	SI (state change)
16	CT16B0	CT16B0 interrupt	Match 0 - 2 Capture 0 - 1
17	CT16B1	CT16B1 interrupt	Match 0 - 1 Capture 0 - 1
18	CT32B0	CT32B0 interrupt	Match 0 - 3 Capture 0 - 1
19	CT32B1	CT32B1 interrupt	Match 0 - 3 Capture 0 -1
20	SSP0	SSP0 interrupt	Tx FIFO half empty Rx FIFO half full Rx Timeout Rx Overrun

Table 6. Connection of interrupt sources to the Vectored Interrupt Controller

Interrupt number	Name	Description	Flag(s)
21	USART0	USART interrupt	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) End of Auto-Baud (ABEO) Auto-Baud Time-Out (ABTO) Modem control interrupt
22	USB_IRQ	USB_IRQ interrupt	USB IRQ interrupt
23	USB_FIQ	USB_FIQ interrupt	USB FIQ interrupt
24	ADC_A	ADC interrupt A	Combined end-of-sequence A and threshold crossing interrupts
25	RTC	RTC interrupt	
26	BOD_WDT	Combined BOD and WWDT interrupt	Brown-out detect and WWDT interrupts
27	FLASH	Flash/EEPROM interrupt	Combined flash and EEPROM controller interrupts
28	DMA	DMA interrupt	
29	ADC_B	ADC interrupt B	Combined end-of-sequence B and overrun interrupts
30	USB_WAKEUP	USB_WAKEUP interrupt	USB wake-up interrupt
31	-	-	Reserved

3.3.3 Non-Maskable Interrupt (NMI)

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

You can set up any peripheral interrupt listed in Table 6 as NMI using the NMISRC register in the SYSCON block (Table 61). 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.

3.3.4 Vector table offset

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

3.4 Register description

See *the ARM Cortex-M0+ technical reference manual*.

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

Table 7. Register overview: NVIC (base address 0xE000 E000)

Name	Access	Address offset	Description	Reset value	Reference
ISER0	R/W	0x100	Interrupt Set Enable Register 0. This register allows enabling interrupts and reading back the interrupt enables for specific peripheral functions.	0	Table 8
-	-	0x104	Reserved.	-	-
ICER0	R/W	0x180	Interrupt Clear Enable Register 0. This register allows disabling interrupts and reading back the interrupt enables for specific peripheral functions.	0	Table 9
-	-	0x184	Reserved.	0	-
ISPR0	R/W	0x200	Interrupt Set Pending Register 0. This register allows changing the interrupt state to pending and reading back the interrupt pending state for specific peripheral functions.	0	Table 10
-	-	0x204	Reserved.	0	-
ICPR0	R/W	0x280	Interrupt Clear Pending Register 0. This register allows changing the interrupt state to not pending and reading back the interrupt pending state for specific peripheral functions.	0	Table 11
-	-	0x284	Reserved.	0	-
IABR0	RO	0x300	Interrupt Active Bit Register 0. This register allows reading the current interrupt active state for specific peripheral functions.	0	Table 12
-	-	0x304	Reserved.	0	-
IPR0	R/W	0x400	Interrupt Priority Registers 0. This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 0 to 3.	0	Table 13
IPR1	R/W	0x404	Interrupt Priority Registers 1 This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 4 to 7.	0	Table 14
IPR2	R/W	0x408	Interrupt Priority Registers 2. This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 8 to 11.	0	Table 15
IPR3	R/W	0x40C	Interrupt Priority Registers 3. This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 12 to 15.	0	Table 16
IPR4	R/W	0x410	Interrupt Priority Registers 4. This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 12 to 15.	0	Table 17
IPR5	R/W	0x414	Interrupt Priority Registers 5. This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 12 to 15.	0	Table 18
IPR6	R/W	0x418	Interrupt Priority Registers 6. This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 24 to 27.	0	Table 19
IPR7	R/W	0x41C	Interrupt Priority Registers 7. This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 28 to 31.	0	Table 20

3.4.1 Interrupt Set Enable Register 0 register

The ISER0 register allows to enable peripheral interrupts or to read the enabled state of those interrupts. Disable interrupts through the ICER0 (Section 3.4.2).

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

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

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

Table 8. Interrupt Set Enable Register 0 register (ISER0, address 0xE000 E100) bit description

Bit	Symbol	Description	Reset value
0	ISE_PININT0	Interrupt enable.	0
1	ISE_PININT1	Interrupt enable.	0
2	ISE_PININT2	Interrupt enable.	0
3	ISE_PININT3	Interrupt enable.	0
4	ISE_PININT4	Interrupt enable.	0
5	ISE_PININT5	Interrupt enable.	0
6	ISE_PININT6	Interrupt enable.	0
7	ISE_PININT7	Interrupt enable.	0
8	ISE_GINT0	Interrupt enable.	0
9	ISE_GINT1	Interrupt enable.	0
10	ISE_I2C1	Interrupt enable.	0
11	ISE_USART1_4	Interrupt enable.	0
12	ISE_USART2_3	Interrupt enable.	0
13	ISE_SCT0_1	Interrupt enable.	0
14	ISE_SSP1	Interrupt enable.	0
15	ISE_I2C0	Interrupt enable.	0
16	ISE_CT16B0	Interrupt enable.	0
17	ISE_CT16B1	Interrupt enable.	0
18	ISE_CT32B0	Interrupt enable.	0
19	ISE_CT32B1	Interrupt enable.	0
20	ISE_SSP0	Interrupt enable.	0
21	ISE_USART0	Interrupt enable.	0
22	ISE_USB_IRQ	Interrupt enable.	0
23	ISE_USB_FIQ	Interrupt enable.	0
24	ISE_ADC_A	Interrupt enable.	0
25	ISE_RTC	Interrupt enable.	0
26	ISE_BOD_WDT	Interrupt enable.	0
27	ISE_FLASH	Interrupt enable.	0
28	ISE_DMA	Interrupt enable.	0
29	ISE_ADC_B	Interrupt enable.	0
30	ISE_USB_WAKEKUP	Interrupt enable.	0
31	-	Reserved	0

3.4.2 Interrupt clear enable register 0

The ICER0 register allows disabling the peripheral interrupts, or for reading the enabled state of those interrupts. Enable interrupts through the ISER0 registers (Section 3.4.1).

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

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

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

Table 9. Interrupt clear enable register 0 (ICER0, address 0xE000 E180)

Bit	Symbol	Description	Reset value
0	ICE_PININT0	Interrupt disable.	0
1	ICE_PININT1	Interrupt disable.	0
2	ICE_PININT2	Interrupt disable.	0
3	ICE_PININT3	Interrupt disable.	0
4	ICE_PININT4	Interrupt disable.	0
5	ICE_PININT5	Interrupt disable.	0
6	ICE_PININT6	Interrupt disable.	0
7	ICE_PININT7	Interrupt disable.	0
8	ICE_GINT0	Interrupt disable.	0
9	ICE_GINT1	Interrupt disable.	0
10	ICE_I2C1	Interrupt disable.	0
11	ICE_USART1_4	Interrupt disable.	0
12	ICE_USART2_3	Interrupt disable.	0
13	ICE_SCT0_1	Interrupt disable.	0
14	ICE_SSP1	Interrupt disable.	0
15	ICE_I2C0	Interrupt disable.	0
16	ICE_CT16B0	Interrupt disable.	0
17	ICE_CT16B1	Interrupt disable.	0
18	ICE_CT32B0	Interrupt disable.	0
19	ICE_CT32B1	Interrupt disable.	0
20	ICE_SSP0	Interrupt disable.	0
21	ICE_USART0	Interrupt disable.	0
22	ICE_USB_IRQ	Interrupt disable.	0
23	ICE_USB_FIQ	Interrupt disable.	0
24	ICE_ADC_A	Interrupt disable.	0
25	ICE_RTC	Interrupt disable.	0
26	ICE_BOD_WDT	Interrupt disable.	0
27	ICE_FLASH	Interrupt disable.	0
28	ICE_DMA	Interrupt disable.	0
29	ICE_ADC_B	Interrupt disable.	0
30	ICE_USB_WAKEUP	Interrupt disable.	0
31	-	Reserved	0

3.4.3 Interrupt Set Pending Register 0 register

The ISPR0 register allows setting the pending state of the peripheral interrupts, or for reading the pending state of those interrupts. Clear the pending state of interrupts through the ICPR0 registers (Section 3.4.4).

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

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

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

Table 10. Interrupt set pending register 0 register (ISPR0, address 0xE000 E200) bit description

Bit	Symbol	Description	Reset value
0	ISP_PININT0	Interrupt pending set.	0
1	ISP_PININT1	Interrupt pending set.	0
2	ISP_PININT2	Interrupt pending set.	0
3	ISP_PININT3	Interrupt pending set.	0
4	ISP_PININT4	Interrupt pending set.	0
5	ISP_PININT5	Interrupt pending set.	0
6	ISP_PININT6	Interrupt pending set.	0
7	ISP_PININT7	Interrupt pending set.	0
8	ISP_GINT0	Interrupt pending set.	0
9	ISP_GINT1	Interrupt pending set.	0
10	ISP_I2C1	Interrupt pending set.	0
11	ISP_USART1_4	Interrupt pending set.	0
12	ISP_USART2_3	Interrupt pending set.	0
13	ISP_SCT0_1	Interrupt pending set.	0
14	ISP_SSP1	Interrupt pending set.	0
15	ISP_I2C0	Interrupt pending set.	0
16	ISP_CT16B0	Interrupt pending set.	0
17	ISP_CT16B1	Interrupt pending set.	0
18	ISP_CT32B0	Interrupt pending set.	0
19	ISP_CT32B1	Interrupt pending set.	0
20	ISP_SSP0	Interrupt pending set.	0
21	ISP_USART0	Interrupt pending set.	0
22	ISP_USB_IRQ	Interrupt pending set.	0
23	ISP_USB_FIQ	Interrupt pending set.	0
24	ISP_ADC_A	Interrupt pending set.	0
25	ISP_RTC	Interrupt pending set.	0
26	ISP_BOD_WDT	Interrupt pending set.	0
27	ISP_FLASH	Interrupt pending set.	0
28	ISP_DMA	Interrupt pending set.	0
29	ISP_ADC_B	Interrupt pending set.	0
30	ISP_USB_WAKEKUP	Interrupt pending set.	0
31	-	Reserved	0

3.4.4 Interrupt Clear Pending Register 0 register

The ICPR0 register allows clearing the pending state of the peripheral interrupts, or for reading the pending state of those interrupts. Set the pending state of interrupts through the ISPR0 register (Section 3.4.3).

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

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

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

Table 11. Interrupt clear pending register 0 register (ICPR0, address 0xE000 E280) bit description

Bit	Symbol	Function	Reset value
0	ICP_PININT0	Interrupt pending clear.	0
1	ICP_PININT1	Interrupt pending clear.	0
2	ICP_PININT2	Interrupt pending clear.	0
3	ICP_PININT3	Interrupt pending clear.	0
4	ICP_PININT4	Interrupt pending clear.	0
5	ICP_PININT5	Interrupt pending clear.	0
6	ICP_PININT6	Interrupt pending clear.	0
7	ICP_PININT7	Interrupt pending clear.	0
8	ICP_GINT0	Interrupt pending clear.	0
9	ICP_GINT1	Interrupt pending clear.	0
10	ICP_I2C1	Interrupt pending clear.	0
11	ICP_USART1_4	Interrupt pending clear.	0
12	ICP_USART2_3	Interrupt pending clear.	0
13	ICP_SCT0_1	Interrupt pending clear.	0
14	ICP_SSP1	Interrupt pending clear.	0
15	ICP_I2C0	Interrupt pending clear.	0
16	ICP_CT16B0	Interrupt pending clear.	0
17	ICP_CT16B1	Interrupt pending clear.	0
18	ICP_CT32B0	Interrupt pending clear.	0
19	ICP_CT32B1	Interrupt pending clear.	0
20	ICP_SSP0	Interrupt pending clear.	0
21	ICP_USART0	Interrupt pending clear.	0
22	ICP_USB_IRQ	Interrupt pending clear.	0
23	ICP_USB_FIQ	Interrupt pending clear.	0
24	ICP_ADC_A	Interrupt pending clear.	0
25	ICP_RTC	Interrupt pending clear.	0
26	ICP_BOD_WDT	Interrupt pending clear.	0
27	ICP_FLASH	Interrupt pending clear.	0
28	ICP_DMA	Interrupt pending clear.	0

Table 11. Interrupt clear pending register 0 register (ICPR0, address 0xE000 E280) bit description *…continued*

Bit	Symbol	Function	Reset value
29	ICP_ADC_B	Interrupt pending clear.	0
30	ICP_USB_WAKEKUP	Interrupt pending clear.	0
31	-	Interrupt pending clear.	0

3.4.5 Interrupt Active Bit Register 0

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

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

Write — n/a.

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

Table 12. Interrupt Active Bit Register 0 (IABR0, address 0xE000 E300) bit description

Bit	Symbol	Function	Reset value
0	IAB_PININT0	Interrupt active state.	0
1	IAB_PININT1	Interrupt active state.	0
2	IAB_PININT2	Interrupt active state.	0
3	IAB_PININT3	Interrupt active state.	0
4	IAB_PININT4	Interrupt active state.	0
5	IAB_PININT5	Interrupt active state.	0
6	IAB_PININT6	Interrupt active state.	0
7	IAB_PININT7	Interrupt active state.	0
8	IAB_GINT0	Interrupt active state.	0
9	IAB_GINT1	Interrupt active state.	0
10	IAB_I2C1	Interrupt active state.	0
11	IAB_USART1_4	Interrupt active state.	0
12	IAB_USART2_3	Interrupt active state.	0
13	IAB_SCT0_1	Interrupt active state.	0
14	IAB_SSP1	Interrupt active state.	0
15	IAB_I2C0	Interrupt active state.	0
16	IAB_CT16B0	Interrupt active state.	0
17	IAB_CT16B1	Interrupt active state.	0
18	IAB_CT32B0	Interrupt active state.	0
19	IAB_CT32B1	Interrupt active state.	0
20	IAB_SSP0	Interrupt active state.	0
21	IAB_USART0	Interrupt active state.	0
22	IAB_USB_IRQ	Interrupt active state.	0
23	IAB_USB_FIQ	Interrupt active state.	0
24	IAB_ADC_A	Interrupt active state.	0
25	IAB_RTC	Interrupt active state.	0

Table 12. Interrupt Active Bit Register 0 (IABR0, address 0xE000 E300) bit description

Bit	Symbol	Function	Reset value
26	IAB_BOD_WDT	Interrupt active state.	0
27	IAB_FLASH	Interrupt active state.	0
28	IAB_DMA	Interrupt active state.	0
29	IAB_ADC_B	Interrupt active state.	0
30	IAB_USB_WAKEKUP	Interrupt active state.	0
31	-	Interrupt active state.	0

3.4.6 Interrupt Priority Register 0

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

Table 13. Interrupt Priority Register 0 (IPR0, address 0xE000 E400) bit description

Bit	Symbol	Description	Reset value
5:0	-	These bits ignore writes, and read as 0.	0
7:6	IP_PIN_INT0	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
13:8	-	These bits ignore writes, and read as 0.	0
15:14	IP_PIN_INT1	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
21:16	-	These bits ignore writes, and read as 0.	0
23:22	IP_PIN_INT2	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
29:24	-	These bits ignore writes, and read as 0.	0
31:30	IP_PIN_INT3	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0

3.4.7 Interrupt Priority Register 1

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

Table 14. Interrupt Priority Register 1 (IPR1, address 0xE000 E404) bit description

Bit	Symbol	Description	Reset value
5:0	-	These bits ignore writes, and read as 0.	0
7:6	IP_PIN_INT4	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
13:8	-	These bits ignore writes, and read as 0.	0
15:14	IP_PIN_INT5	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
21:16	-	These bits ignore writes, and read as 0.	0
23:22	IP_PIN_INT6	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
29:24	-	These bits ignore writes, and read as 0.	0
31:30	IP_PIN_INT7	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0

3.4.8 Interrupt Priority Register 2

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

Table 15. Interrupt Priority Register 2 (IPR2, address 0xE000 E408) bit description

Bit	Symbol	Description	Reset value
5:0	-	These bits ignore writes, and read as 0.	0
7:6	IP_GINT0	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
13:8	-	These bits ignore writes, and read as 0.	0
15:14	IP_GINT1	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
21:16	-	These bits ignore writes, and read as 0.	0
23:22	IP_I2C1	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
29:24	-	These bits ignore writes, and read as 0.	0
31:30	IP_USART1_4	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0

3.4.9 Interrupt Priority Register 3

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

Table 16. Interrupt Priority Register 3 (IPR3, address 0xE000 E40C) bit description

Bit	Symbol	Description	Reset value
5:0	-	These bits ignore writes, and read as 0.	0
7:6	IP_USART2_3	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
13:8	-	These bits ignore writes, and read as 0.	0
15:14	IP_SCT0_1	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
21:16	-	These bits ignore writes, and read as 0.	0
23:22	IP_SSP1	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
29:24	-	These bits ignore writes, and read as 0.	0
31:30	IP_I2C0	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0

3.4.10 Interrupt Priority Register 4

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

Table 17. Interrupt Priority Register 4 (IPR4, address 0xE000 E410) bit description

Bit	Symbol	Description	Reset value
5:0	-	These bits ignore writes, and read as 0.	0
7:6	IP_CT16B0	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
13:8	-	These bits ignore writes, and read as 0.	0
15:14	IP_CT16B1	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
21:16	-	These bits ignore writes, and read as 0.	0
23:22	IP_CT32B0	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
29:24	-	These bits ignore writes, and read as 0.	0
31:30	IP_CT32B1	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0

3.4.11 Interrupt Priority Register 5

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

Table 18. Interrupt Priority Register 5 (IPR5, address 0xE000 E414) bit description

Bit	Symbol	Description	Reset value
5:0	-	These bits ignore writes, and read as 0.	0
7:6	IP_SSP0	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
13:8	-	These bits ignore writes, and read as 0.	0
15:14	IP_USART0	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
21:16	-	These bits ignore writes, and read as 0.	0
23:22	IP_USB_IRQ	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0
29:24	-	These bits ignore writes, and read as 0.	0
31:30	IP_USB_FIQ	Interrupt Priority. 0 = highest priority. 3 = lowest priority.	0

3.4.12 Interrupt Priority Register 6

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

Table 19. Interrupt Priority Register 6 (IPR6, address 0xE000 E418) bit description

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

3.4.13 Interrupt Priority Register 7

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

Table 20. Interrupt Priority Register 7 (IPR7, address 0xE000 E41C) bit description

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

UM10732

Chapter 4: LPC11U6x/E6x System configuration (SYSCON)

Rev. 1.8 — 19 December 2016 **User manual**

4.1 How to use this chapter

The SYSCON block is identical for all parts. USB and USB PLL related registers are available on LPC11U6x only and are reserved on LPC11E6x. The USB PLL is only available on the LPC11U6x.

4.2 Basic configuration

No clock configuration is needed. The clock to the SYSCON block is always enabled.

By default, the SYSCON block is clocked by the IRC.

4.2.1 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 4.4.48 "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.

   Section 4.4.12 "System PLL clock source select register"
3. Update the PLL clock source in the SYSPLLCLKUEN register.

   Section 4.4.13 "System PLL clock source update register"
4. Configure the PLL M and N dividers.

   Section 4.4.3 "System PLL control register"
5. Wait for the PLL to lock by monitoring the PLL lock status.

   Section 4.4.4 "System PLL status register"

4.2.2 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, from the PLL, or directly from the 32 kHz (32.768 kHz) oscillator.

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. See Section 4.2.1 “Set up the PLL”.
  - 32 kHz clock: set the source for the PLL input to the 32 kHz clock in the SYSPLLCLKSEL register and select PLL input in the MAINCLKSEL register. The 32 kHz oscillator output must be also enabled in the RTCOSCCTRL register.

   Section 4.4.16 “Main clock source select register”

2. Update the main clock source.

   Section 4.4.17 “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 4.4.18 “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 4.4.19 “System clock control register”

4.2.3 Set up the system oscillator using XTALIN and XTALOUT

To use the system oscillator, you need to enable the XTALIN and XTALOUT pins through the IOCON registers.

1. In the IOCON block, disable the pull-up and pull-down resistors in the IOCON registers for pins PIO2_0 and PIO2_1 and set the MODE bits to 0x1.
2. In the SYSOSCCTRL register, disable the BYPASS bit and select the oscillator frequency range according to the desired oscillator output clock.

Related registers:

Table 89 “Digital/analog pin control registers (PIO2_[0:1], addresses 0x4004 40F0 (PIO2_0) to 0x4004 40F4 (PIO2_1)) bit description”

Table 29 “System oscillator control (SYSOSCCTRL, address 0x4004 8020) bit description”

4.3 General description

4.3.1 Clock generation

The system control block generates all clocks for the chip. Except for the USART clocks, the SSP clocks, 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 50 MHz. See Figure 4.

Each clock divider can either disable the clock or divide the clock by values between 1 and 255. Therefore, the peripheral clocks to the SSPs, UARTs, and IOCON can run at frequencies different from the system clock frequency. The USB clock can be either generated by a dedicated PLL or derived from the main clock. For low-speed USB, the IRC with 1 % accuracy can be selected as the USB clock source. See Section 15.4.8 “USB Low-speed operation”.

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

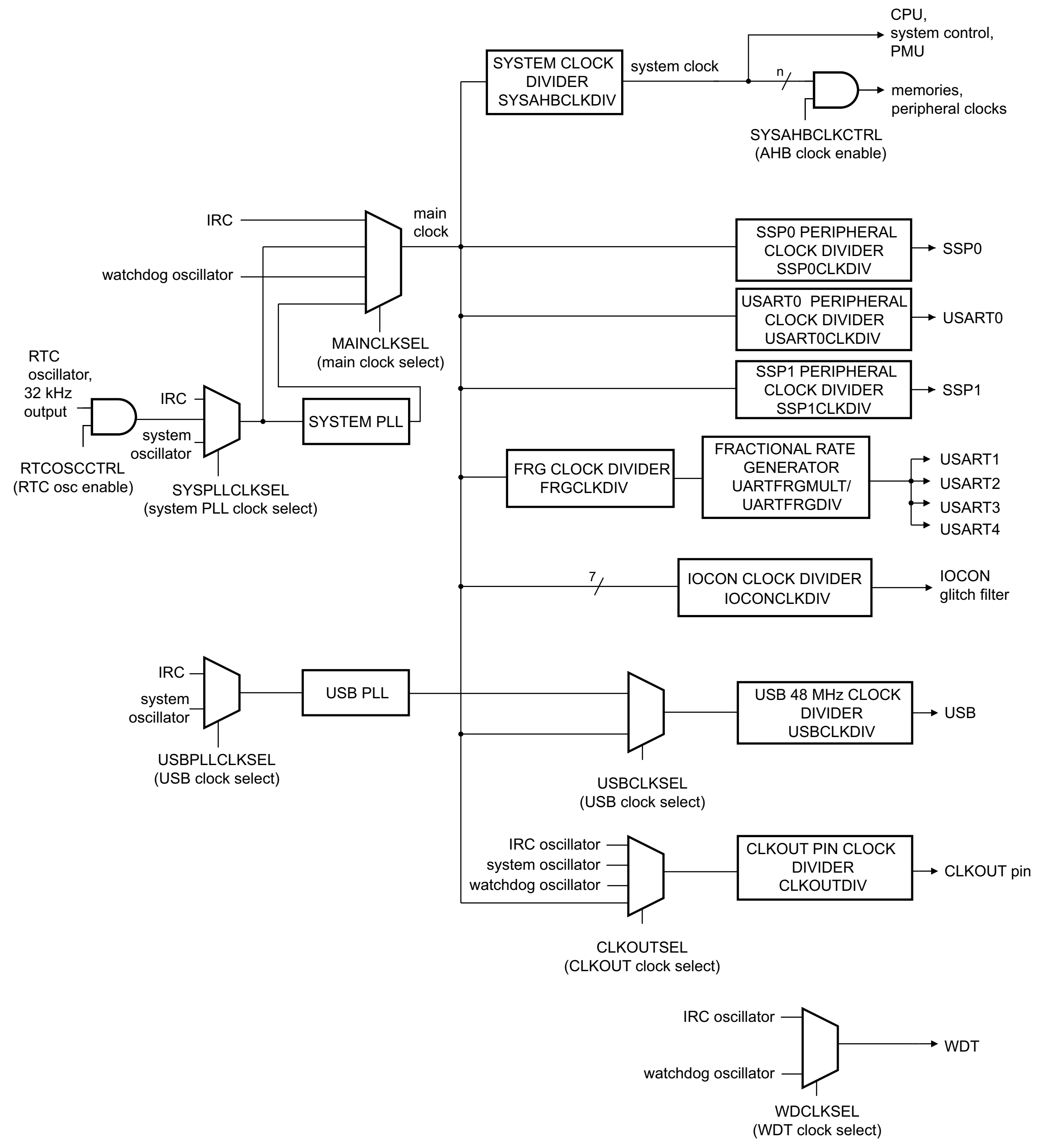


Fig 4. Clock generation

4.3.2 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 temperature sensor. For details, see the following registers:

Section 4.4.46 “Deep-sleep mode configuration register”

Section 4.4.3 “System PLL control register”

Section 4.4.9 “Watchdog oscillator control register”

Section 4.4.8 “System oscillator control register”

4.3.3 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 4.4.48 “Power configuration register”

Section 4.4.45 “Start logic 1 interrupt wake-up enable register”

4.3.4 Reset and interrupt control

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

Up to eight external pin interrupts can be assigned to any digital pin except PIO2_8 to PIO2_23 in the system control block (see Section 4.4.41 “Pin Interrupt Select registers 0 to 7”).

4.4 Register description

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

Name	Access	Address offset	Description	Reset value	Reset value after boot	Reference
SYSMEMREMAP	R/W	0x000	System memory remap	0		Table 22
PRESETCTRL	R/W	0x004	Peripheral reset control	0		Table 23
SYSPLLCTRL	R/W	0x008	System PLL control	0		Table 24
SYSPLLSTAT	R	0x00C	System PLL status	0		Table 25
USBPLLCTRL	R/W	0x010	USB PLL control	0		Table 26
USBPLLSTAT	R	0x014	USB PLL status	0		Table 27
RTCOSCCTRL	R/W	0x01C	RTC oscillator 32 kHz output control	0x1		Table 28
SYSOSCCTRL	R/W	0x020	System oscillator control	0x000		Table 29
WDTOSCCTRL	R/W	0x024	Watchdog oscillator control	0	0	Table 30
IRCCTRL	R/W	0x028	IRC control	0x080	-	Table 31
SYSRSTSTAT	R/W	0x030	System reset status register	0		Table 32
SYSPLLCLKSEL	R/W	0x040	System PLL clock source select	0		Table 33
SYSPLLCLKUEN	R/W	0x044	System PLL clock source update enable	0x1	0x1	Table 34
USBPLLCLKSEL	R/W	0x048	USB PLL clock source select	0		Table 35

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

Name	Access	Address offset	Description	Reset value	Reset value after boot	Reference
USBPLLCLKUEN	R/W	0x04C	USB PLL clock source update enable	0	0	Table 36
MAINCLKSEL	R/W	0x070	Main clock source select	0		Table 37
MAINCLKUEN	R/W	0x074	Main clock source update enable	0x1	0x1	Table 38
SYSAHBCLKDIV	R/W	0x078	System clock divider	0x001		Table 39
SYSAHBCLKCTRL	R/W	0x080	System clock control	0x3F	0x800 4857	Table 40
SSP0CLKDIV	R/W	0x094	SSP0 clock divider	0		Table 41
USART0CLKDIV	R/W	0x098	USART0 clock divider	0		Table 42
SSP1CLKDIV	R/W	0x09C	SSP1 clock divider	0x0000		Table 43
FRGCLKDIV	R/W	0x0A0	Clock divider for the common fractional baud rate generator of USART1, USART2, USART3, USART4	0		Table 44
-	-	0x0AC	-	-		-
-	-	0x0B0	-	-		-
USBCLKSEL	R/W	0x0C0	USB clock source select	0		Table 45
USBCLKUEN	R/W	0x0C4	USB clock source update enable	0	0	Table 46
USBCLKDIV	R/W	0x0C8	USB clock source divider	0		Table 47
CLKOUTSEL	R/W	0x0E0	CLKOUT clock source select	0		Table 48
CLKOUTUEN	R/W	0x0E4	CLKOUT clock source update enable	0	0	Table 49
CLKOUTDIV	R/W	0x0E8	CLKOUT clock divider	0		Table 50
UARTFRGDIV	R/W	0x0F0	USART fractional generator divider value	0		Table 51
UARTFRGMULT	R/W	0x0F4	USART fractional generator multiplier value	0		Table 52
EXTTRACECMD	R/W	0x0FC	External trace buffer command register	0		Table 53
PIOPORCAP0	R	0x100	POR captured PIO status 0	user dependent		Table 54
PIOPORCAP1	R	0x104	POR captured PIO status 1	user dependent		Table 55
PIOPORCAP2	R	0x108	POR captured PIO status 1	user dependent		Table 55
IOCONCLKDIV6	R/W	0x134	Peripheral clock 6 to the IOCON block for programmable glitch filter	0x0000 0000		Table 57
IOCONCLKDIV5	R/W	0x138	Peripheral clock 5 to the IOCON block for programmable glitch filter	0x0000 0000		Table 57
IOCONCLKDIV4	R/W	0x13C	Peripheral clock 4 to the IOCON block for programmable glitch filter	0x0000 0000		Table 57
IOCONCLKDIV3	R/W	0x140	Peripheral clock 3 to the IOCON block for programmable glitch filter	0x0000 0000		Table 57
IOCONCLKDIV2	R/W	0x144	Peripheral clock 2 to the IOCON block for programmable glitch filter	0x0000 0000		Table 57

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

Name	Access	Address offset	Description	Reset value	Reset value after boot	Reference
IOCONCLKDIV1	R/W	0x148	Peripheral clock 1 to the IOCON block for programmable glitch filter	0x0000 0000		Table 57
IOCONCLKDIV0	R/W	0x14C	Peripheral clock 0 to the IOCON block for programmable glitch filter	0x0000 0000		Table 57
BODCTRL	R/W	0x150	Brown-Out Detect	0		Table 58
SYSTCKCAL	R/W	0x154	System tick counter calibration			Table 59
-	-	0x158 - 0x16C	Reserved	-	-	-
IRQLATENCY	R/W	0x170	IRQ delay. Allows trade-off between interrupt latency and determinism.	0x0000 0010		Table 60
NMISRC	R/W	0x174	NMI Source Control	0		Table 61
PINTSEL0	R/W	0x178	GPIO Pin Interrupt Select register 0	0		Table 62
PINTSEL1	R/W	0x17C	GPIO Pin Interrupt Select register 1	0		Table 62
PINTSEL2	R/W	0x180	GPIO Pin Interrupt Select register 2	0		Table 62
PINTSEL3	R/W	0x184	GPIO Pin Interrupt Select register 3	0		Table 62
PINTSEL4	R/W	0x188	GPIO Pin Interrupt Select register 4	0		Table 62
PINTSEL5	R/W	0x18C	GPIO Pin Interrupt Select register 5	0		Table 62
PINTSEL6	R/W	0x190	GPIO Pin Interrupt Select register 6	0		Table 62
PINTSEL7	R/W	0x194	GPIO Pin Interrupt Select register 7	0		Table 62
USBCLKCTRL	R/W	0x198	USB clock control			Table 63
USBCLKST	R	0x19C	USB clock status			Table 64
STARTERP0	R/W	0x204	Start logic 0 interrupt wake-up enable register 0	0		Table 65
STARTERP1	R/W	0x214	Start logic 1 interrupt wake-up enable register 1	0		Table 66
PDSLEEPCFG	R/W	0x230	Power-down states in deep-sleep mode			Table 67
PDAWAKECFG	R/W	0x234	Power-down states for wake-up from deep-sleep			Table 68
PDRUNCFG	R/W	0x238	Power configuration register			Table 69
DEVICE_ID	R	0x3F4	Device ID	part dependent		Table 70

4.4.1 System memory remap register

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

Table 22. System memory remap (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	-

4.4.2 Peripheral reset control register

This register allows software to reset specific peripherals. Writing a 0 to an assigned bit in this register resets the specified peripheral. Writing a 1 negates the reset and allows peripheral operation.

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

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

Bit	Symbol	Value	Description	Reset value
0	SSP0_RST_N		SSP0 reset control	0
		0	Reset. Resets the SSP0 peripheral.	
		1	Clear reset. SSP0 reset de-asserted.	
1	I2C0_RST_N		I2C0 reset control	0
		0	Reset. Resets the I2C0 peripheral.	
		1	Clear reset. I2C0 reset de-asserted.	
2	SSP1_RST_N		SSP1 reset control	0
		0	Reset. Resets the SSP1 peripheral.	
		1	Clear reset. SSP1 reset de-asserted.	
3	I2C1_RST_N		I2C1 reset control	0
		0	Reset. Resets the I2C1 peripheral.	
		1	Clear reset. I2C1 reset de-asserted.	
4	FRG_RST_N		FRG reset control	0
		0	Reset. Resets the FRG peripheral.	
		1	Clear reset. FRG reset de-asserted.	
5	USART1_RST_N		USART1 reset control	0
		0	Reset. Resets the USART1 peripheral.	
		1	Clear reset. USART1 reset de-asserted.	
6	USART2_RST_N		USART2 reset control	0
		0	Reset. Resets the USART2 peripheral.	
		1	Clear reset. USART2 reset de-asserted.	

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

Bit	Symbol	Value	Description	Reset value
7	USART3_RST_N		USART3 reset control	0
		0	Reset. Resets the USART3 peripheral.	
		1	Clear reset. USART3 reset de-asserted.	
8	USART4_RST_N		USART4 reset control	0
		0	Reset. Resets the USART4 peripheral.	
		1	Clear reset. USART4 reset de-asserted.	
9	SCT0_RST_N		SCT0 reset control	0
		0	Reset. Resets the SCT0 peripheral.	
		1	Clear reset. SCT0 reset de-asserted.	
10	SCT1_RST_N		SCT1 reset control	0
		0	Reset. Resets the SCT1 peripheral.	
		1	Clear reset. SCT1 reset de-asserted.	
31:11	-		Reserved	-

4.4.3 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.

Table 24. System PLL control (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 x P.	0
		0x0	P = 1	
		0x1	P = 2	
		0x2	P = 4	
		0x3	P = 8	
31:7	-		Reserved. Do not write ones to reserved bits.	-

4.4.4 System PLL status register

This register is a Read-only register and supplies the PLL lock status.

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

Bit	Symbol	Value	Description	Reset value
0	LOCK		PLL lock status	0
		0	No lock. PLL not locked	
		1	Lock. PLL locked	
31:1	-		Reserved	-

4.4.5 USB PLL control register

The USB PLL is identical to the system PLL and is used to provide a dedicated clock to the USB block.

This register connects and enables the USB PLL and configures the PLL multiplier and divider values. The PLL accepts an input frequency from 10 MHz to 25 MHz from various clock sources. The input frequency is multiplied up to a high frequency, then divided down to provide the actual clock 48 MHz clock used by the USB subsystem.

Table 26. USB PLL control (USBPLLCTRL, address 0x4004 8010) bit description

Bit	Symbol	Value	Description	Reset value
4:0	MSEL		Feedback divider value. The division value M is the programmed MSEL value + 1. 00000: Division ratio M = 1 to 11111: Division ratio M = 32	0x000
6:5	PSEL		Post divider ratio P. The division ratio is 2 x P.	0x00
		0x0	P = 1	
		0x1	P = 2	
		0x2	P = 4	
		0x3	P = 8	
31:7	-		Reserved. Do not write ones to reserved bits.	0x00

4.4.6 USB PLL status register

This register is a Read-only register and supplies the PLL lock status.

Table 27. USB PLL status (USBPLLSTAT, address 0x4004 8014) bit description

Bit	Symbol	Value	Description	Reset value
0	LOCK		PLL lock status	0x0
		0	No lock. PLL not locked	
		1	Lock. PLL locked	
31:1	-		Reserved	0x00

4.4.7 RTC oscillator 32 kHz output control register

This register enables the 32 kHz (32.768 kHz) output of the RTC oscillator. The 32 kHz clock can be used to create a very slow main clock by selecting the 32 kHz as the system PLL clock and then using the PLL input as the clock source to the main clock. Do not use the system PLL with 32 kHz clock.

Table 28. RTC oscillator 32 kHz output control (RTCOSCCTRL, address 0x4004 801C) bit description

Bit	Symbol	Value	Description	Reset value
0	RTCOSCEN		Enable the RTC 32 kHz output.	1
		0	Disabled. 32 kHz output disabled.	
		1	Enabled. 32 kHz output enabled.	
31:1	-		Reserved	-

4.4.8 System oscillator control register

This register configures the frequency range for the system oscillator. The system oscillator itself is powered on or off in the PDRUNCFG register. See Table 69.

Table 29. System oscillator control (SYSOSCCTRL, address 0x4004 8020) bit description

Bit	Symbol	Value	Description	Reset value
0	BYPASS		Bypass system oscillator	0x0
		0	Oscillator is not bypassed.	
		1	Bypass enabled. PLL input (sys_osc_clk) is fed directly from the XTALIN 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	Low. 1 - 20 MHz frequency range.	
		1	High. 15 - 25 MHz frequency range.	
31:2	-		Reserved	0x00

4.4.9 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)) = 9.4 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 30. 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

4.4.10 Internal resonant crystal control register

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

Table 31. Internal resonant crystal control register (IRCCTRL, address 0x4004 8028) bit description

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

4.4.11 System reset status register

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

Table 32. 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	
1	EXTRST		Status of the external $\overline{\text{RESET}}$ pin	0
		0	No reset event detected	
		1	Reset detected	
2	WDT		Status of the Watchdog reset	0
		0	No WDT reset detected	
		1	WDT reset detected	
3	BOD		Status of the Brown-out detect reset	0
		0	No BOD reset detected	
		1	BOD reset detected	
4	SYSRST		Status of the software system reset	0
		0	No System reset detected	
		1	System reset detected	
31:5	-		Reserved	-

4.4.12 System PLL clock source select register

This register selects the clock source for the system PLL. The output of this clock select register can also be used as the source of the main clock without using the PLL (pll input option in the MAINCLKSEL register).

Table 33. System PLL clock source select (SYSPLLCLKSEL, address 0x4004 8040) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		System PLL clock source	0
		0x0	IRC	
		0x1	System oscillator. Crystal Oscillator (SYSOSC)	
		0x2	Reserved	
		0x3	32 kHz clock.Select this option when the 32 kHz clock is the clock source for the main clock and select the pll input in the MAINCLKSEL register. Do not use the 32 kHz clock with the PLL.	
31:2	-		Reserved	-

4.4.13 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 34. 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	1
		0	No change	
		1	Update clock source	
31:1	-	-	Reserved	-

4.4.14 USB PLL clock source select register

This register selects the clock source for the dedicated USB PLL.

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

Table 35. USB PLL clock source select (USBPLLCLKSEL, address 0x4004 8048) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		USB PLL clock source	0x00
		0x0	IRC. For full-speed USB, switch the USB PLL clock source to the system oscillator for correct USB operation. The IRC is suitable for low-speed USB operation only.	
		0x1	System oscillator	
		0x2	Reserved	
		0x3	Reserved	
31:2	-		Reserved	0x00

4.4.15 USB PLL clock source update enable register

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

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

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

Table 36. USB PLL clock source update enable register (USBPLLCLKUEN, address 0x4004 804C) bit description

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

4.4.16 Main clock source select register

This register selects the main system clock, which can be the system PLL output (sys_pllclkout), the PLL input (to connect the 32 kHz clock to the main clock), the watchdog oscillator, or the IRC oscillator. The main system clock clocks the core, the peripherals, and the memories.

Table 37. Main clock source select (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	-

4.4.17 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 38. 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	1
		0	No change	
		1	Update clock source	
31:1	-	-	Reserved	-

4.4.18 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 39. System clock divider (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	-

4.4.19 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 40. System clock control (SYSAHBCLKCTRL, address 0x4004 8080) bit description

Bit	Symbol	Value	Description	Reset value
0	SYS		This bit is read-only and always reads as 1. It configures the always-on clock for the AHB, the APB bridges, the Cortex-M0 core clocks, SYSCON, reset control, SRAM0, and the PMU. Writes to this bit are ignored.	1
1	ROM		Enables clock for ROM.	1
		0	Disable	
		1	Enable	
2	RAM0		Enables clock for Main SRAM0.	1
		0	Disable	
		1	Enable	
3	FLASHREG		Enables clock for flash register interface.	1
		0	Disabled	
		1	Enabled	
4	FLASHARRAY		Enables clock for flash access.	1
		0	Disabled	
		1	Enabled	
5	I2C0		Enables clock for I2C.	0
		0	Disable	
		1	Enable	
6	GPIO		Enables clock for GPIO port registers.	1
		0	Disable	
		1	Enable	
7	CT16B0		Enables clock for 16-bit counter/timer 0.	0
		0	Disable	
		1	Enable	
8	CT16B1		Enables clock for 16-bit counter/timer 1.	0
		0	Disable	
		1	Enable	

Table 40. System clock control (SYSAHBCLKCTRL, address 0x4004 8080) bit description

Bit	Symbol	Value	Description	Reset value
9	CT32B0		Enables clock for 32-bit counter/timer 0.	0
		0	Disable	
		1	Enable	
10	CT32B1		Enables clock for 32-bit counter/timer 1.	0
		0	Disable	
		1	Enable	
11	SSP0		Enables clock for SSP0.	1
		0	Disable	
		1	Enable	
12	USART0		Enables clock for USART0.	0
		0	Disable	
		1	Enable	
13	ADC		Enables clock for ADC.	0
		0	Disable	
		1	Enable	
14	USB		Enables clock to the USB register interface.	1
		0	Disable	
		1	Enable	
15	WWDT		Enables clock for WWDT.	0
		0	Disable	
		1	Enable	
16	IOCON		Enables clock for I/O configuration block.	0
		0	Disable	
		1	Enable	
17	-		Reserved	0
18	SSP1		Enables clock for SSP1.	0
		0	Disable	
		1	Enable	
19	PINT		Enables clock to GPIO Pin interrupt register interface.	0
		0	Disable	
		1	Enable	
20	USART1		Enables clock to USART1 register interface.	0
		0	Disable	
		1	Enable	
21	USART2		Enables clock to USART2 register interface.	0
		0	Disable	
		1	Enable	

Table 40. System clock control (SYSAHBCLKCTRL, address 0x4004 8080) bit description

Bit	Symbol	Value	Description	Reset value
22	USART3_4		Enables clock to USART3 and USART4 register interfaces.	0
		0	Disable	
		1	Enable	
23	GROUP0INT		Enables clock to GPIO GROUP0 interrupt register interface.	0
		0	Disable	
		1	Enable	
24	GROUP1INT		Enables clock to GPIO GROUP1 interrupt register interface.	0
		0	Disable	
		1	Enable	
25	I2C1		Enables clock for I2C1.	0
		0	Disable	
		1	Enable	
26	RAM1		Enables clock for SRAM1 located at 0x2000 0000 to 0x2000 0800.	0
		0	Disable	
		1	Enable	
27	USBSRAM		Enables USB SRAM/SRAM2 block located at 0x2000 4000 to 0x2000 4800.	1
		0	Disable	
		1	Enable	
28	CRC		Enables clock for CRC.	0
		0	Disable	
		1	Enable	
29	DMA		Enables clock for DMA.	0
		0	Disable	
		1	Enable	
30	RTC		Enables clock for RTC register interface.	0
		0	Disable	
		1	Enable	
31	SCT0_1		Enables clock for SCT0 and SCT1.	0
		0	Disable	
		1	Enable	

4.4.20 SSP0 clock divider register

This register configures the SSP0 peripheral clock SPI0_PCLK. SPI0_PCLK can be shut down by setting the DIV field to zero.

Table 41. SSP0 clock divider (SSP0CLKDIV, address 0x4004 8094) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	SPI0_PCLK clock divider values. 0: System clock disabled. 1: Divide by 1. to 255: Divide by 255.	0
31:8	-	Reserved	-

4.4.21 USART0 clock divider register

This register configures the USART peripheral clock UART_PCLK. The UART_PCLK can be shut down by setting the DIV field to zero.

Table 42. USART0 clock divider (USART0CLKDIV, address 0x4004 8098) bit description

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

4.4.22 SSP1 clock divider register (SSP1CLKDIV)

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

Table 43. SSP1 clock divider (SSP1CLKDIV, address 0x4004 809C) bit description

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

4.4.23 UART Fractional baud rate clock divider register

This register configures the clock for the fractional baud rate generator and USART1 to USART4. The USART clock can be disabled by setting the DIV field to zero (this is the default setting).

Remark: This register does not configure the clock to the USART0 peripheral. See Table 42.

Table 44. UART Fractional baud rate clock divider register (FRGCLKDIV, address 0x4004 80A0) 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	-

4.4.24 USB clock source select register (USBCLKSEL)

This register selects the clock source for the USB usb_clk. The clock source can be either the USB PLL output or the main clock, and the clock can be further divided by the USBCLKDIV register (see Table 47) to obtain a 48 MHz clock.

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

Table 45. USB clock source select (USBCLKSEL, address 0x4004 80C0) bit description

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

4.4.25 USB clock source update enable register

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

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

Table 46. USB clock source update enable register (USBCLKUEN, address 0x4004 80C4) bit description

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

4.4.26 USB clock source divider register (USBCLKDIV)

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

Table 47. USB clock source divider (USBCLKDIV, address 0x4004 80C8) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	USB clock divider values 0: Disable USB clock. 1: Divide by 1. to 255: Divide by 255.	0x01
31:8	-	Reserved	0x00

4.4.27 CLKOUT clock source select register (CLKOUTSEL)

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

To change the clock source visible on the CLKOUT pin, first enable the new clock source with the currently selected clock source still running, change the clock source using the SEL bit, and then remove the current clock source.

If the clock source selected on the CLKOUT pin is powered down in the PDRUNCFG or PDSLEEPCFG registers, this same clock source must be re-enabled before another clock source can be selected through this register.

Table 48. CLKOUT clock source select (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

4.4.28 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 49. 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	-

4.4.29 CLKOUT clock divider register (CLKOUTDIV)

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

Table 50. CLKOUT clock divider (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	-

4.4.30 USART fractional generator divider value register

The USART1 to USART4 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 FRGCLKDIV 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 52.

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 51. 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	-

4.4.31 USART fractional generator multiplier value register

The USART1 to USART4 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 FRGCLKDIV 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 51.
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 52. 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	-

4.4.32 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 53. 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

4.4.33 POR captured PIO status 0 register

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 54. POR captured PIO status 0 (PIOPORCAP0, address 0x4004 8100) bit description

Bit	Symbol	Description	Reset value
23:0	PIOSTAT	State of PIO0_23 through PIO0_0 at power-on reset	Implementation dependent
31:24	-	Reserved	-

4.4.34 POR captured PIO status 1 register

The PIOPORCAP1 register captures the state of GPIO port 1 at power-on-reset. Each bit represents the reset state of one GPIO pin. This register is a read-only status register.

Table 55. POR captured PIO status 1 (PIOPORCAP1, address 0x4004 8104) bit description

Bit	Symbol	Description	Reset value
31:0	PIOSTAT	State of PIO1_31 through PIO1_0 at power-on reset	Implementation dependent

4.4.35 POR captured PIO status 2 register

The PIOPORCAP2 register captures the state of GPIO port 2 at power-on-reset. Each bit represents the reset state of one GPIO pin. This register is a read-only status register.

Table 56. POR captured PIO status 2 (PIOPORCAP2, address 0x4004 8108) bit description

Bit	Symbol	Description	Reset value
23:0	PIOSTAT	State of PIO2_23 through PIO2_0 at power-on reset	Implementation dependent

4.4.36 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 57. 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

4.4.37 Brown-Out Detect register

The BOD control register selects up to four separate threshold values for sending a BOD interrupt to the NVIC and for forced reset. Reset and interrupt threshold values listed in Table 58 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 Table 75.

Table 58. Brown-Out Detect (BODCTRL, address 0x4004 8150) bit description

Bit	Symbol	Value	Description	Reset value
1:0	BODRSTLEV		BOD reset level	00
		0x0	Level 0.	
		0x1	Level 1.	
		0x2	Level 2.	
		0x3	Level 3.	
3:2	BODINTVAL		BOD interrupt level	00
		0x0	Reserved.	
		0x1	Reserved	
		0x2	Level 2.	
		0x3	Level 3.	
4	BODRSTENA		BOD reset enable	0
		0	Disable reset function.	
		1	Enable reset function.	
31:5	-		Reserved	0x00

4.4.38 System tick counter calibration register

This register determines the value of the SYST_CALIB register (see Table 352).

Table 59. System tick counter calibration (SYSTCKCAL, address 0x4004 8154) bit description

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

4.4.39 IRQ delay register

The IRQLATENCY register is an 8-bit register which specifies the minimum number of cycles (0-255) permitted for the system to respond to an interrupt request. The intent of this register is to allow the user to select a trade-off between interrupt response time and determinism.

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

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

The default setting for this register is 0x010.

Table 60. IRQ delay (IRQLATENCY, address 0x4004 8170) bit description

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

4.4.40 NMI Source Control register

The NMI source selection register selects a peripheral interrupts 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 6. For a description of the NMI functionality, see *ARM Cortex-M0 technical reference manual*.

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 61. NMI Source Control (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 6 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, as described in the *ARM Cortex-M0 technical reference manual*.

4.4.41 Pin Interrupt Select registers 0 to 7

The pin interrupt select register is an input mux for the pin interrupt and allows to select any pin (except PIO2_8 to PIO2_23) as an external interrupt. A total of eight pin interrupt are supported. Each of the eight PINTSEL registers selects one GPIO pin from the following pins as external pin interrupt:

- Port 0: PIO0_0 to PIO0_23 (Pin number INTPIN = 0 to 23)
- Port 1: PIO1_0 to PIO1_31 (Pin number INTPIN = 24 to 55)
- Port 2: PIO2_0 to PIO2_7 (Pin number INTPIN = 56 to 63)

The selected pin of each PINTSEL register is connected to the corresponding pin interrupt in the NVIC. The pin interrupt must be enabled using interrupt slots # 0 to 7 (see Table 6).

To enable each pin interrupt and configure its edge or level sensitivity, use the GPIO pin interrupt registers (see Table 106).

Table 62. GPIO Pin Interrupt Select registers (PINTSEL[0:7], address 0x4004 8178 (PINTSEL0) to 0x4004 8194 (PINTSEL7)) bit description

Bit	Symbol	Value	Description	Reset value
5:0	INTPIN		Pin number. PIO0_0 = 0, ..., PIO0_23 = 23, PIO1_0 = 24, ..., PIO1_31 = 55, PIO2_0 = 56, ..., PIO2_7 = 63.	0
31:6	-		Reserved	-

4.4.42 USB clock control register

This register controls the use of the USB need_clock signal and the polarity of the need_clock signal for triggering the USB wake-up interrupt. For details of how to use the USB need_clock signal for waking up the part from Deep-sleep or Power-down modes, see Section 15.3.1.

Table 63. USB clock control (USBCLKCTRL, address 0x4004 8198) bit description

Bit	Symbol	Value	Description	Reset value
0	AP_CLK		USB need_clock signal control	0x0
		0	Hardware. Under hardware control.	
		1	Forced. Forced HIGH.	
1	POL_CLK		USB need_clock polarity for triggering the USB wake-up interrupt	0x0
		0	Falling edge. Falling edge of the USB need_clock triggers the USB wake-up (default).	
		1	Rising edge. Rising edge of the USB need_clock triggers the USB wake-up.	
2	-		Reserved. Only write 0 to this bit.	
31:3	-		Reserved	0x00

4.4.43 USB clock status register

This register is read-only and returns the status of the USB need_clock signal. For details of how to use the USB need_clock signal for waking up the part from Deep-sleep or Power-down modes, see Section 15.3.1.

Table 64. USB clock status (USBCLKST, address 0x4004 819C) bit description

Bit	Symbol	Value	Description	Reset value
0	NEED_CLKST		USB need_clock signal status	0x0
		0	LOW	
		1	HIGH	
31:1	-		Reserved	0x00

4.4.44 Start logic 0 interrupt wake-up enable register 0

The STARTERP0 register enables the individual GPIO pins selected through the Pin interrupt select registers (see Table 62) for wake-up. The pin interrupts must also be enabled in the NVIC (interrupts 0 to 8 in Table 6).

Table 65. Start logic 0 interrupt wake-up enable register 0 (STARTERP0, address 0x4004 8204) bit description

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

4.4.45 Start logic 1 interrupt wake-up enable register

This register selects which interrupts will wake the part from deep-sleep and power-down modes. Interrupts selected by a one in these registers must be enabled in the NVIC (Table 6).

The STARTERP1 register enables the WWDT interrupt, the BOD interrupt, the USB wake-up interrupt and the two GPIO group interrupts for wake-up.

Table 66. Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description

Bit	Symbol	Value	Description	Reset value
11:0			Reserved.	-

Table 66. Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description

Bit	Symbol	Value	Description	Reset value
12	RTCINT		RTC interrupt wake-up	0
		0	Disabled	
		1	Enabled	
13	WWDT_BODINT		Combined WWDT interrupt or Brown Out Detect (BOD) interrupt wake-up	0
		0	Disabled	
		1	Enabled	
18:14	-		Reserved	-
19	USB_WAKEUP		USB need_clock signal wake-up	0
		0	Disabled	
		1	Enabled	
20	GROUP0INT		GPIO GROUP0 interrupt wake-up	0
		0	Disabled	
		1	Enabled	
21	GROUP1INT		GPIO GROUP1 interrupt wake-up	0
		0	Disabled	
		1	Enabled	
22	-		Reserved.	0
23	USART1_4		Combined USART1 and USART4 interrupt wake-up	0
		0	Disabled	
		1	Enabled	
24	USART2_3		Combined USART2 and USART3 interrupt wake-up	0
		0	Disabled	
		1	Enabled	
31:25			Reserved.	-

4.4.46 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 exception of 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 335) is set. See Section 22.4.4 for details.

Table 67. Deep-sleep mode 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
		1	Powered down	
		0	Powered	
5:4	-		Reserved.	0b11
6	WDTOSC_PD		Watchdog oscillator power-down control for Deep-sleep and Power-down mode	1
		1	Powered down	
		0	Powered	
31:7	-		Reserved	-

4.4.47 Wake-up configuration register

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

Table 68. Wake-up configuration (PDAWAKECFG, address 0x4004 8234) bit description

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

Table 68. Wake-up configuration (PDAWAKECFG, address 0x4004 8234) bit description

Bit	Symbol	Value	Description	Reset value
7	SYSPLL_PD		System PLL wake-up configuration	1
		1	Powered down	
		0	Powered	
8	USBPLL_PD		USB PLL wake-up configuration	1
		0	Powered	
		1	Powered down	
9	-		Reserved. Always write this bit as 0.	
10	USBPAD_PD		USB transceiver wake-up configuration	1
		0	Powered	
		1	Powered down	
11	-		Reserved. This bit must be set to one in Run mode.	1
12	-		Reserved.	0
13	TEMPSENSE_PD		Temperature sensor wake-up configuration	1
		0	Powered	
		1	Powered down	
31:14	-		Reserved	-

4.4.48 Power configuration register

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

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

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

Table 69. Power configuration register (PDRUNCFG, address 0x4004 8238) bit description

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

Table 69. Power configuration register (PDRUNCFG, address 0x4004 8238) bit description

Bit	Symbol	Value	Description	Reset value
2	FLASH_PD		Flash power-down	0
		0	Powered	
		1	Powered down	
3	BOD_PD		BOD power-down	0
		0	Powered	
		1	Powered down	
4	ADC_PD		ADC power-down	1
		0	Powered	
		1	Powered down	
5	SYSOSC_PD		Crystal oscillator power-down. After power-up, add a software delay of approximately 500 μs before using.	1
		0	Powered	
		1	Powered down	
6	WDTOSC_PD		Watchdog oscillator power-down	1
		0	Powered	
		1	Powered down	
7	SYSPLL_PD		System PLL power-down	1
		0	Powered	
		1	Powered down	
8	USBPLL_PD		USB PLL power-down	1
		0	Powered	
		1	Powered down	
9	-		Reserved. Always write this bit as 0.	
10	USBPAD_PD		USB transceiver power-down configuration	1
		0	Powered	
		1	Powered down	
11	-		Reserved. This bit must be set to one in Run mode.	1
12	-		Reserved.	0
13	TEMPSENSE_PD		Temperature sensor wake-up configuration	1
		0	Powered	
		1	Powered down	
15:14	-		Reserved. **Always write these bits as 0b11.**	0b11
31:16	-		Reserved	-

4.4.49 Device ID register

This device ID register is a read-only register and contains the part ID for each part. This register is also read by the ISP/IAP commands (see Table 376).

LPC11U67JBD48 = 0x0000 BC88

LPC11U68JBD48 = 0x0000 7C08

LPC11U68JBD64 = 0x0000 7C08

LPC11U68JBD100 = 0x0000 7C00

LPC11U67JBD100 = 0x0000 BC80

LPC11U67JBD64 = 0x0000 BC88

LPC11U66JBD48 = 0x0000 DCC8

LPC11E67JBD48 = 0x0000 BC81

LPC11E68JBD64 = 0x0000 7C01

LPC11E68JBD100 = 0x0000 7C01

LPC11E68JBD48 = 0x0000 7C01

LPC11E67JBD100 = 0x0000 BC81

LPC11E67JBD64 = 0x0000 BC81

LPC11E66JBD48 = 0x0000 DCC1

Table 70. Device ID (DEVICE_ID, address 0x400483F8) bit description

Bit	Symbol	Description	Reset value
31:0	DEVICEID	PARTID	part-dependent

4.5 Functional description

4.5.1 Reset

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

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

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

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

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

4.5.2 Start-up behavior

See Figure 5 for the start-up timing after reset. The IRC is the default clock at Reset and provides a clean system clock shortly after the supply voltage reaches the threshold value of 1.8 V.

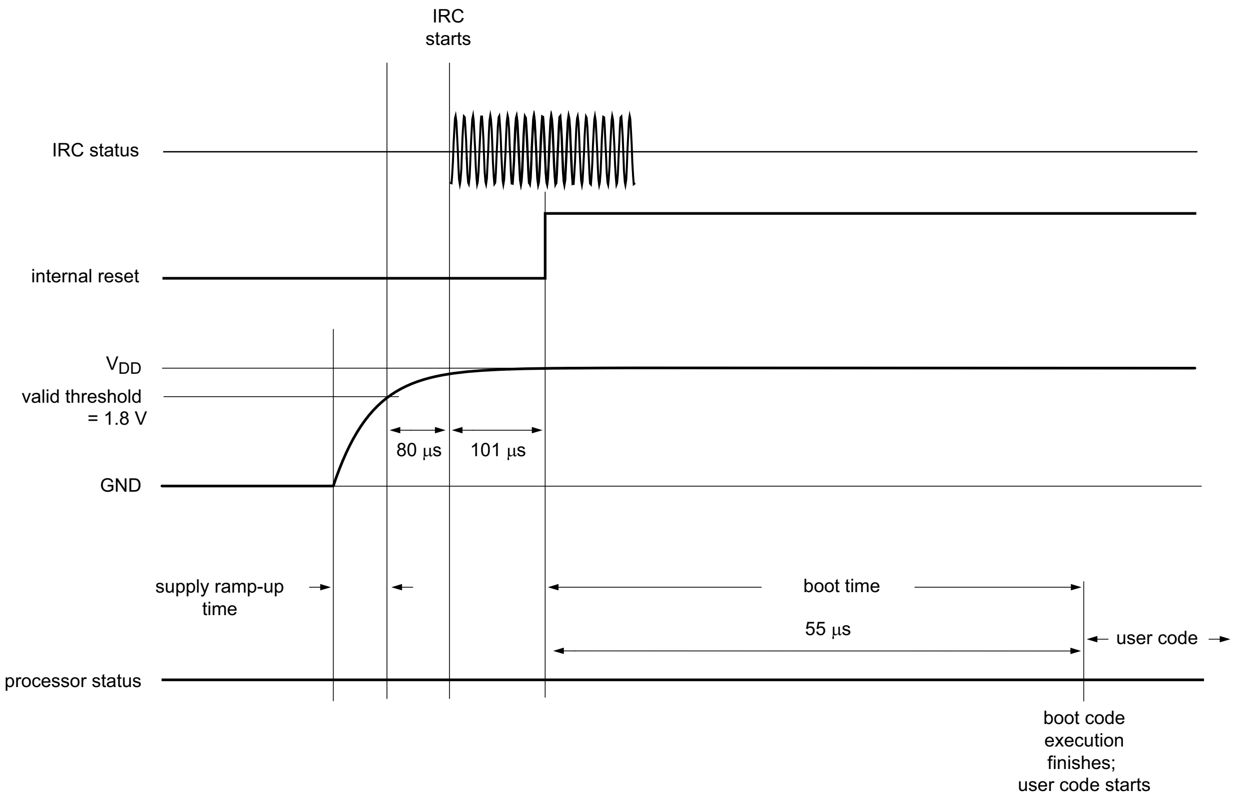


Fig 5. Start-up timing

4.5.3 Brown-out detection

The part includes up to four levels for monitoring the voltage on the V_{DD} pin. If this voltage falls below one of the selected levels, the BOD asserts an interrupt signal to the NVIC or issues a reset, depending on the value of the BODRSTENA bit in the BOD control register (Table 58).

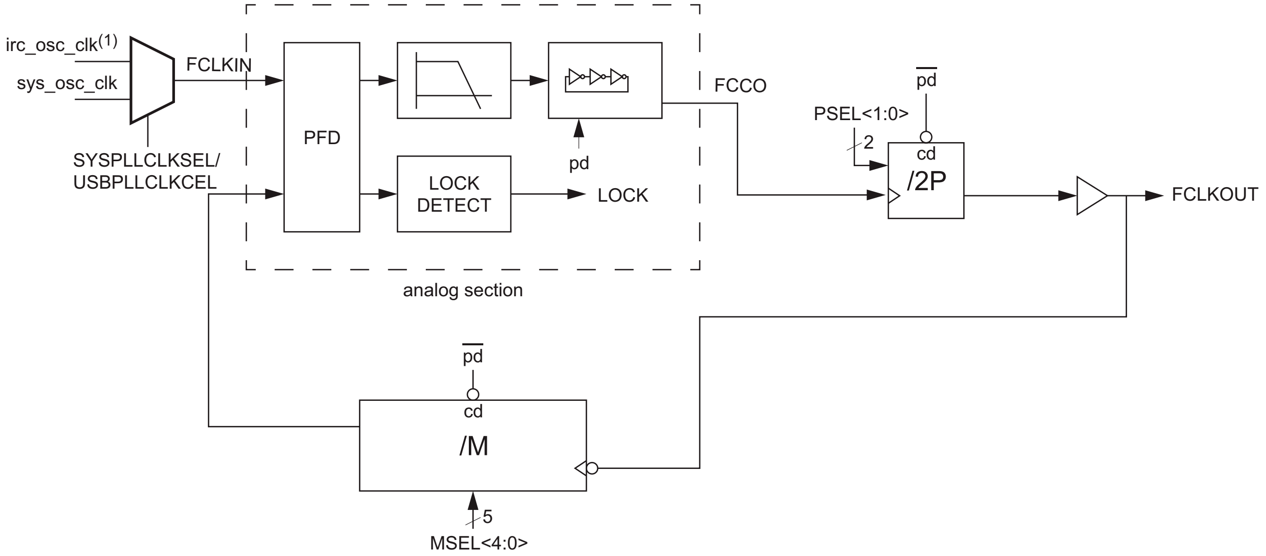
The interrupt signal can be enabled for interrupt in the Interrupt Enable Register in the NVIC (see Table 7) in order to cause a CPU interrupt; if not, software can monitor the signal by reading a dedicated status register.

If the BOD interrupt is enabled in the STARTERP1 register (see Table 66) and in the NVIC, the BOD interrupt can wake up the chip from Deep-sleep and power-down mode.

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

4.5.4 System PLL/USB PLL functional description

The part uses the system PLL to create the clocks for the core and peripherals. An identical PLL is available for the USB.



(1) System PLL only. For USB PLL, use IRC with XTAL-less low-speed USB operation only.

Fig 6. System PLL block diagram

The block diagram of this PLL is shown in Figure 6. 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.

4.5.4.1 Lock detector

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

4.5.4.2 Power-down control

To reduce the power consumption when the PLL clock is not needed, a Power-down mode has been incorporated. This mode is enabled by setting the SYSPLL_PD bit to one in the Power-down configuration register (Table 69). In this mode, the internal current reference will be turned off, the oscillator and the phase-frequency detector will be stopped and the dividers will enter a reset state. While in Power-down mode, the lock output will be low to indicate that the PLL is not in lock. When the Power-down mode is terminated by setting the SYSPLL_PD bit to zero, the PLL will resume its normal operation and will make the lock signal high once it has regained lock on the input clock.

4.5.4.3 Divider ratio programming

Post divider

The division ratio of the post divider is controlled by the PSEL bits. The division ratio is two times the value of P selected by PSEL bits as shown in Table 24 and Table 26. This guarantees an output clock with a 50% duty cycle.

Feedback divider

The feedback divider’s division ratio is controlled by the MSEL bits. The division ratio between the PLL’s output clock and the input clock is the decimal value on MSEL bits plus one, as specified in Table 24 and Table 26.

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.

4.5.4.4 Frequency selection

The PLL frequency equations use the following parameters (also see Figure 6):

Table 71. PLL frequency parameters

Parameter	System PLL
FCLKIN	Frequency of sys_pllclkin (input clock to the system PLL) from the SYSPLLCLKSEL multiplexer (see Table 33).
FCCO	Frequency of the Current Controlled Oscillator (CCO); 156 to 320 MHz.
FCLKOUT	Frequency of sys_pllclkout
P	System PLL post divider ratio; PSEL bits in SYSPLLCTRL (see Table 24).
M	System PLL feedback divider register; MSEL bits in SYSPLLCTRL (see Table 24).

4.5.4.4.1 Normal mode

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

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

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

1. Specify the input clock frequency Fclkin.
2. Calculate M to obtain the desired output frequency Fclkout with M = F_{clkout} / F_{clkin}.
3. Find a value so that FCCO = 2 × P × F_{clkout}.
4. Verify that all frequencies and divider values conform to the limits specified in Table 24 and Table 26.

Table 72 shows how to configure the PLL for a 12 MHz crystal oscillator using the SYSPLLCTRL register (Table 24). The main clock is equivalent to the system clock if the system clock divider SYSAHBCLKDIV is set to one (see Table 39).

Table 72. PLL configuration examples

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

4.5.4.4.2 Power-down mode

In this mode, the internal current reference will be turned off, the oscillator and the phase-frequency detector will be stopped and the dividers will enter a reset state. While in Power-down mode, the lock output will be low, to indicate that the PLL is not in lock. When the Power-down mode is terminated by setting the SYSPLL_PD or USB_PLL bit to zero in the Power-down configuration register (Table 69), the PLL will resume its normal operation and will make the lock signal high once it has regained lock on the input clock.

UM10732

Chapter 5: LPC11U6x/E6x Power Management Unit (PMU)

Rev. 1.8 — 19 December 2016 User manual

5.1 How to read this chapter

The PMU is identical for all parts.

5.2 Basic configuration

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

5.2.1 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.

5.2.1.1 System control register

The System control register (SCR) controls entry to and exit from a low power state. This register is located on the private peripheral bus and is a R/W register with reset value of 0x0000 0000. The SCR register allows to put the ARM core into sleep mode or the entire system in Deep-sleep or Power-down mode. To set the low power state with SLEEPDEEP = 1 to either deep-sleep or power-down or to enter the Deep power-down mode, use the PCON register (Table 77).

Table 73. 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

Table 73. System control register (SCR, address 0xE000 ED10) bit description

Bit	Symbol	Description	Reset value
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

5.3 General description

Power 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 as configured.

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 and BOD can remain active to wake up the system on an interrupt. If the RTC is running, it can wake up the part.

   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 and BOD can remain active to wake up the system on an interrupt. If the RTC is running, it can wake up the part.

3. Deep power-down mode:

   For maximal power savings, the entire system is shut down except for the general purpose registers in the PMU, the RTC in the VBAT power domain, and the WAKEUP pin if V_{DD} is present. Only the general purpose registers in the PMU and the RTC registers are powered and can maintain their internal states. The part can wake up on a pulse on the WAKEUP pin or on an interrupt from the RTC. On wake-up, the part boots.

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

5.3.1 Peripheral configuration in the reduced power modes

All peripherals which can cause the part to wake up are configurable to achieve the lowest possible power consumption while the part is in reduced power mode. Only peripherals that are needed to generate an interrupt in sleep mode or a wake-up signal should be enabled.

Table 74. 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
USART1/2/3/4	software configurable	off; but can create wake-up interrupt in synchronous slave mode or 32 kHz clock mode	off; but can create wake-up interrupt in synchronous slave mode or 32 kHz clock mode	off
RTC	software configurable	software configurable	software configurable	software configurable
Other peripherals	software configurable	off	off	off

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

Remark: USART0 cannot be used to wake up the part from deep-sleep or power-down modes.

5.3.2 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 68 “Wake-up configuration (PDAWAKECFG, address 0x4004 8234) bit description”.
- To use external interrupts for wake-up: Table 65 “Start logic 0 interrupt wake-up enable register 0 (STARTERP0, address 0x4004 8204) bit description” and Table 62 “GPIO 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 66 “Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description”
- To configure the USART to wake up the part: Section 11.4.2 “Configure the USART for wake-up”
- For configuring the RTC: Section 21.4
- For a list of all wake-up sources: Table 75 “Wake-up sources for reduced power modes”

Table 75. 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.
	$\overline{\text{RESET}}$ pin	The RESET function must be selected in the IOCON block. This is the default.
	RTC 1 kHz timer time-out and alarm	• Enable interrupt in NVIC and STARTERP1 registers. • Enable 1 Hz or 1 kHz RTC clock. • Configure the RTC WAKEUP timer.
	Interrupt from USART1/2/3/4 peripherals	• Enable interrupt in NVIC and STARTERP1 registers. • Enable USART1/2/3/4 interrupts. • Provide an external clock signal to the peripheral. • Configure the USART in synchronous slave mode. • 32 kHz mode if the 32 kHz oscillator is connected. To use the 32 kHz as a low-power system clock for the USART1/2/3/4, also enable the 32 kHz signal in the RTCOSCCTRL.
	USB wake-up interrupt	• Enable interrupt in NVIC and STARTERP1 registers. • Configure the USBCLKCTRL register.
Deep power-down	WAKEUP pin PIO0_16	Enable the WAKEUP function in the GPREG4 register in the PMU. V_{DD} must be present.
	RTC 1 kHz timer time-out and alarm	• Enable the RTC oscillator in the RTC CTRL register. • Enable RTC alarm signal to wake up the part from Deep power-down mode in the RTC CTRL register. • Select RTC clock in the RTC CTRL register. • Start RTC wake-up timer by writing a time-out value to the RTC WAKE register.

5.4 Register description

Table 76. Register overview: PMU (base address 0x4003 8000)

Name	Access	Address offset	Description	Reset value	Reference
PCON	R/W	0x000	Power control register	0x0	Table 77
GPREG0	R/W	0x004	General purpose register 0	0x0	Table 78
GPREG1	R/W	0x008	General purpose register 1	0x0	Table 78
GPREG2	R/W	0x00C	General purpose register 2	0x0	Table 78
GPREG3	R/W	0x010	General purpose register 3	0x0	Table 78
GPREG4	R/W	0x014	General purpose register 4. Deep power-down control register.	0x0	Table 79

5.4.1 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. See Section 5.5 for details on how to enter the power-down modes.

Table 77. Power control register (PCON, address 0x4003 8000) bit description

Bit	Symbol	Value	Description	Reset value
2:0	PM		Power mode	000
		0x0	Default. The part is in active or sleep mode.	
		0x1	Deep-sleep. ARM WFI will enter Deep-sleep mode.	
		0x2	Power-down. ARM WFI will enter Power-down mode.	
		0x3	Deep power-down. ARM WFI will enter Deep-power down mode (ARM Cortex-M0+ core powered-down).	
3	NODPD		A 1 in this bit prevents entry to Deep power-down mode when 0x3 is written to the PM field above, the SLEEPDEEP bit is set, and a WFI is executed. This bit is cleared only by power-on reset, so writing a one to this bit locks the part in a mode in which Deep power-down mode is blocked.	0
7:4	-	-	Reserved. Do not write ones to this bit.	0
8	SLEEPFLAG		Sleep mode flag	0
		0	Active mode. Read: No power-down mode entered. Part is in Active mode. Write: No effect.	
		1	Low power mode. Read: Sleep/deep-sleep or power-down mode entered. Write: Writing a 1 clears the SLEEPFLAG bit to 0.	
10:9	-	-	Reserved. Do not write ones to this bit.	0

Table 77. Power control register (PCON, address 0x4003 8000) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
11	DPDFLAG		Deep power-down flag	0
		0	Not entered. Read: Deep power-down mode **not** entered. Write: No effect.	0
		1	Entered. Read: Deep power-down mode entered. Write: Clear the Deep power-down flag.	
31:12	-	-	Reserved. Do not write ones to this bit.	0

5.4.2 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 78. General purpose registers 0 to 3 (GPREG[0:3], address 0x4003 8004 (GPREG0) to 0x4003 8010 (GPREG3)) bit description

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

5.4.3 General purpose register 4/Deep power-down control

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

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

Table 79. General purpose register 4 (GPREG4, address 0x4003 8014) bit description

Bit	Symbol	Value	Description	Reset value
9:0	-	-	Reserved. Do not write ones to this bit.	0x0
10	WAKEUPHYS		WAKEUP pin hysteresis enable	0x0
		0	Disable Hysteresis for WAKUP pin disabled.	
		1	Enable. Hysteresis for WAKEUP pin enabled.	

Table 79. General purpose register 4 (GPREG4, address 0x4003 8014) bit description

Bit	Symbol	Value	Description	Reset value
11	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 should only disable the wake-up pin if the RTC wake-up timer is enabled and configured to wake up the part. **Remark:** Setting this bit is not necessary if Deep power-down mode is not used.	0
		0	Enable. The wake-up function is enabled on pin PIO0_16.	
		1	Disable. Setting this bit disables the wake-up function on pin PIO0_16.	
31:12	GPDATA		Data retained during Deep power-down mode.	0x0

5.5 Functional description

5.5.1 Power management

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

5.5.2 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 22.4.4 “Using the WWDT lock features”.

5.5.3 Active mode

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

The 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.

5.5.3.1 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 40).

- The power to various analog blocks (PLL, oscillators, the ADC, the BOD circuit, and the flash block) can be controlled at any time individually through the PDRUNCFG register (Table 69).
- The clock source for the system clock can be selected from the IRC (default), the system oscillator, or the watchdog oscillator (see Figure 4 and related registers).
- The system clock frequency can be selected by the SYSPLLCTRL (Table 24) and the SYSAHBCLKDIV register (Table 39).
- Selected peripherals (USART, SSP0/1, USB, CLKOUT) use individual peripheral clocks with their own clock dividers. The peripheral clocks can be shut down through the corresponding clock divider registers (Table 44 to Table 51).

5.5.4 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.

5.5.4.1 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.

5.5.4.2 Programming Sleep mode

The following steps must be performed to enter Sleep mode:

1. The PD 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.
3. Use the ARM Cortex-M0+ Wait-For-Interrupt (WFI) instruction.

5.5.4.3 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.

5.5.5 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 must 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.

Remark: If the LOCK bit is set in the WWDT MOD register (Table 335) and the IRC is selected as a clock source for the WWDT, the IRC continues to clock the WWDT in Deep-sleep 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.

5.5.5.1 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 67) register:

- The watchdog oscillator can be left running in Deep-sleep mode if required for the WWDT.
- If the IRC is locked as the WWDT clock source (see Section 22.4.4), the IRC continues to run and clock the WWDT in Deep-sleep mode independently of the setting in the PDSLEEPCFG register.
- The BOD circuit can be left running in Deep-sleep mode if required by the application.

5.5.5.2 Programming Deep-sleep mode

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

1. The PD bits in the PCON register must be set to 0x1 (Table 77).
2. Select the power configuration in Deep-sleep mode in the PDSLEEPCFG (Table 67) register.
3. Determine if the WWDT clock source must be locked to override the power configuration in case the IRC is selected as clock for the WWDT (see Section 22.4.4).
4. If the main clock is not the IRC, power up the IRC in the PDRUNCFG register and switch the clock source to IRC in the MAINCLKSEL register (Table 37). This ensures that the system clock is shut down glitch-free.
5. Select the power configuration after wake-up in the PDAWAKECFG (Table 68) register.
6. If any of the available wake-up interrupts are needed for wake-up, enable the interrupts in the interrupt wake-up registers (Table 65, Table 66) and in the NVIC.
7. Write one to the SLEEPDEEP bit in the ARM Cortex-M0+ SCR register.
8. Use the ARM WFI instruction.

5.5.5.3 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 62. Each pin interrupt must also be enabled in the STARTERP0 register (Table 65) 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 66). 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 58).
- WWDT signal, if the watchdog oscillator is enabled in the PDSLEEPCFG register:
  - WWDT interrupt using the interrupt wake-up register 1 (Table 66). 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. 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.
- USB wake-up signal using the interrupt wake-up register 1 (Table 66). For details, see Section 15.3.1.
- GPIO group interrupt signal. The interrupt must also be enabled in the STARTERP1 register (Table 66) and in the NVIC.
- RTC alarm signal or wake-up signal. See Section 21.3. Interrupt must also be enabled in the STARTERP1 register (Table 66) and in the NVIC.

Remark: If the watchdog oscillator is running in Deep-sleep mode, its frequency determines the wake-up time.

Remark: If the application in active mode uses a main clock different from the IRC, reprogram the clock source for the main clock in the MAINCLKSEL register after waking up.

5.5.6 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.

Remark: Do not set the LOCK bit in the WWDT MOD register (Table 335) when the IRC is selected as a clock source for the WWDT. This prevents the part from entering the Power-down mode correctly.

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.

5.5.6.1 Power configuration in Power-down mode

Power consumption in Power-down mode can be configured by the power configuration setting in the PDSLEEPCFG (Table 67) register in the same way as for Deep-sleep mode (see Section 5.5.5.1):

- 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.

5.5.6.2 Programming Power-down mode

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

1. The PD bits in the PCON register must be set to 0x2 (Table 77).
2. Select the power configuration in Power-down mode in the PDSLEEPCFG (Table 67) register.
3. If the lock bit 5 in the WWDT MOD register is set (Table 335) and the IRC is selected as the WWDT clock source, reset the part to clear the lock bit and then select the watchdog oscillator as the WWDT clock source.
4. If the main clock is not the IRC, power up the IRC in the PDRUNCFG register and switch the clock source to IRC in the MAINCLKSEL register (Table 37). This ensures that the system clock is shut down glitch-free.
5. Select the power configuration after wake-up in the PDAWAKECFG (Table 69) register.
6. If any of the available wake-up interrupts are used for wake-up, enable the interrupts in the interrupt wake-up registers (Table 65, Table 66) and in the NVIC.
7. Write one to the SLEEPDEEP bit in the ARM Cortex-M0+ SCR register.
8. Use the ARM WFI instruction.

5.5.6.3 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 62. Each pin interrupt must also be enabled in the STARTERP0 register (Table 65) 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 66). 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 58).
- WWDT signal, if the watchdog oscillator is enabled in the PDSLEEPCFG register:
  - WWDT interrupt using the interrupt wake-up register 1 (Table 66). 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.

- USB wake-up signal interrupt wake-up register 1 (Table 66). For details, see Section 15.3.1.
- GPIO group interrupt signal. The interrupt must also be enabled in the STARTERP1 register (Table 66) and in the NVIC.
- RTC alarm signal or wake-up signal. See Section 21.3. Interrupt must also be enabled in the STARTERP1 register (Table 66) and in the NVIC.

Remark: If the watchdog oscillator is running in mode, its frequency determines the wake-up time.

Remark: If the application in active mode uses a main clock different from the IRC, reprogram the clock source for the main clock in the MAINCLKSEL register after waking up.

5.5.7 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. The Deep power-down mode is controlled by the PMU.

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.

Remark: Setting bit 3 in the PCON register (Section 5.4.1) prevents the part from entering Deep-power down mode.

5.5.7.1 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 is powered.

5.5.7.2 Programming Deep power-down mode

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

1. Pull the WAKEUP pin externally HIGH.
2. Ensure that bit 3 in the PCON register (Table 77) is cleared.
3. Write 0x3 to the PD bits in the PCON register (see Table 77).
4. Store data to be retained in the general purpose registers (Section 5.4.2).
5. Write one to the SLEEPDEEP bit in the ARM Cortex-M0 SCR register.
6. Use the ARM WFI instruction.

5.5.7.3 Wake-up from Deep power-down mode

Pulling the WAKEUP pin LOW wakes up the part from Deep power-down, and the chip goes through the entire reset process (Section 4.5.1).

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 GPREG0 to GPREG4 and PCON will be in their reset state.
2. Once the chip has booted, read the deep power-down flag in the PCON register (Table 77) 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 77).
4. (Optional) Read the stored data in the general purpose registers (Section 5.4.2).
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.

5.5.7.4 Programming Deep power-down mode using the RTC for wake-up:

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

1. Set up the RTC high-resolution timer. Write to the RTC VAL register. This starts the high-resolution timer if enabled. Another option is to use the 1 Hz alarm timer.
2. Ensure that bit 3 in the PCON register (Table 77) is cleared.
3. Store data to be retained in the general purpose registers (Section 5.4.2).
4. Use the ARM WFI instruction.

5.5.7.5 Wake-up from Deep power-down mode using the RTC:

The part goes through the entire reset process when the RTC times out:

1. When the high-resolution timer 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 boots.
   - All registers except the GPREG0 to GPREG4 registers and PCON will be in their reset state.
2. Once the chip has booted, read the deep power-down flag in the PCON register (Table 77) 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 77).
4. (Optional) Read the stored data in the general purpose registers (Section 5.4.2).
5. Set up the PMU for the next Deep power-down cycle.

UM10732

Chapter 6: LPC11U6x/E6x I/O control (IOCON)

Rev. 1.8 — 19 December 2016 **User manual**

6.1 How to read this chapter

The IOCON register map depends on the package type (see Table 80). Registers for pins which are not pinned out are reserved. USB-related pin functions are only available on the LPC11U6x parts.

Table 80. IOCON registers available

Package	Port 0	Port 1	Port 2
LPC11U6x			
LQFP48	PIO0_0 to PIO0_23	PIO1_13, PIO1_20, PIO1_21, PIO1_23, PIO1_24	PIO2_0 to PIO2_2, PIO2_5, PIO2_7
LQFP64	PIO0_0 to PIO0_23	PIO1_0, PIO1_7, PIO1_9, PIO1_10, PIO1_13, PIO1_19 to PIO1_21, PIO1_23, PIO1_24, PIO1_26 to PIO1_30	PIO2_0 to PIO2_2, PIO2_5 to PIO2_7, PIO2_15, PIO2_18, PIO2_19
LQFP100	PIO0_0 to PIO0_23	PIO1_0 to PIO1_31	PIO2_0 to PIO2_23
LPC11E6x			
LQFP48	PIO0_0 to PIO0_23	PIO1_13, PIO1_20, PIO1_21, PIO1_23, PIO1_24	PIO2_0 to PIO2_2, PIO2_3, PIO2_4, PIO2_5, PIO2_7
LQFP64	PIO0_0 to PIO0_23	PIO1_0, PIO1_7, PIO1_9, PIO1_10, PIO1_13, PIO1_19 to PIO1_21, PIO1_23, PIO1_24, PIO1_26 to PIO1_30	PIO2_0 to PIO2_2, PIO2_3, PIO2_4, PIO2_5 to PIO2_7, PIO2_15, PIO2_18, PIO2_19
LQFP100	PIO0_0 to PIO0_23	PIO1_0 to PIO1_31	PIO2_0 to PIO2_23

6.2 Features

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

- Pin function
- Internal pull-up/pull-down resistor or bus keeper function (repeater mode)
- Open-drain mode for standard I/O pins
- Hysteresis
- Input inverter
- Glitch filter on selected pins
- Analog input or digital mode for pads hosting the ADC inputs
- I^{2}C mode for pads hosting the I^{2}C-bus function

6.3 Basic configuration

Enable the clock to the IOCON in the SYSAHBCLKCTRL register (Table 40). Once the pins are configured, you can disable the IOCON clock to conserve power.

Each pin has a programmable digital input filter. The base clock for the filter is the output of the IOCONCLKDIV clock divider in the SYSCON block (see Table 57). The base clock can be divided individually for each pin to create the glitch filter constant in each digital pin configuration register.

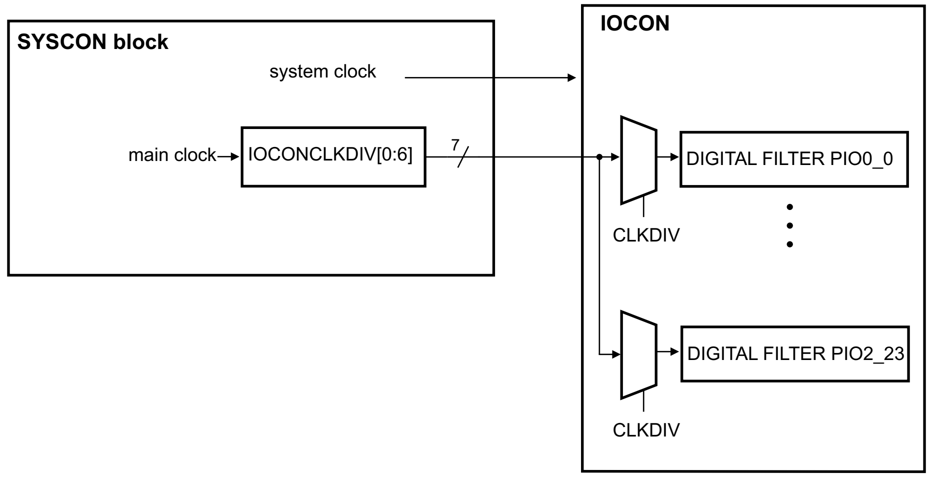


Fig 7. IOCON clocking

6.4 General description

The IOCON registers control the function (GPIO or peripheral function) and the electrical characteristics of the port pins (see Figure 8).

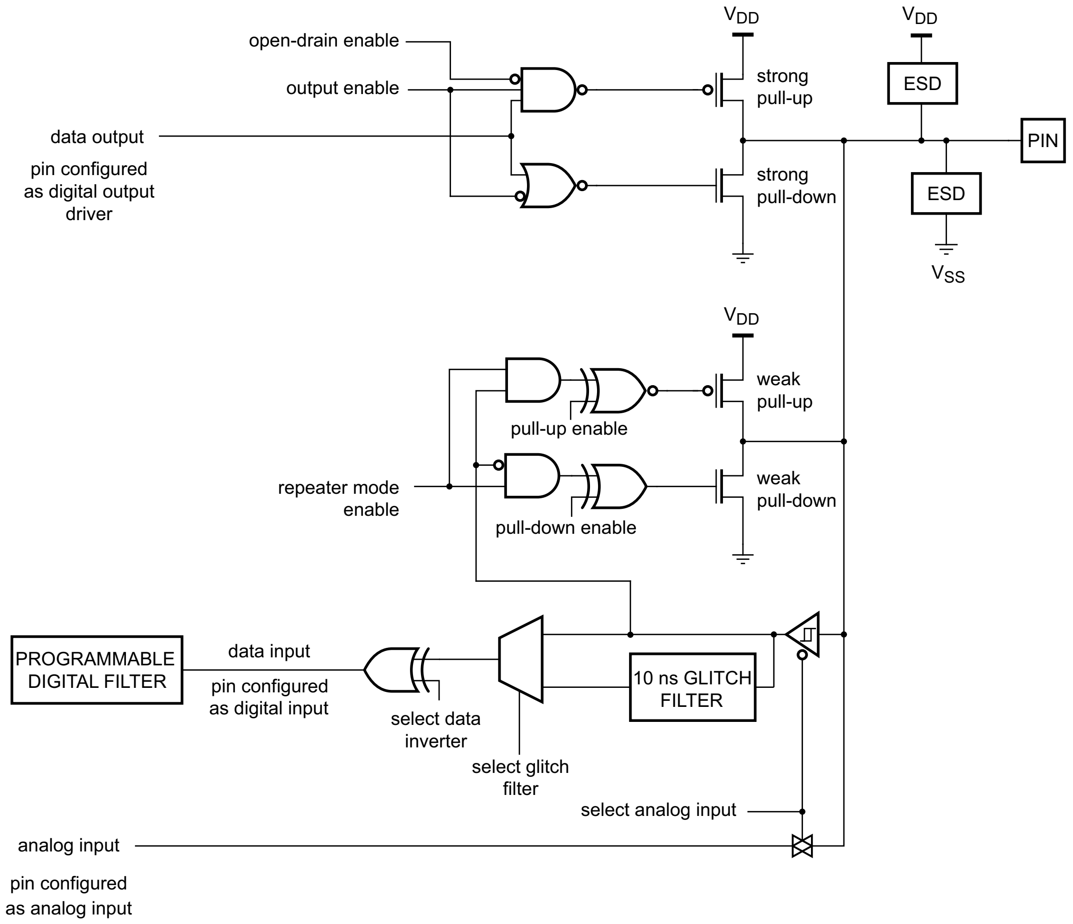


Fig 8. Standard I/O pin configuration

6.4.1 Pin function

The FUNC bits in the IOCON registers can be set to GPIO (FUNC = 000) or to a peripheral function (see Table 83 "IOCON function assignments"). If the pins are GPIO pins, the DIR registers determine whether the pin is configured as an input or output (see Table 95). For any peripheral function, the pin direction is controlled automatically depending on the pin's functionality. The DIR registers have no effect for peripheral functions.

6.4.2 Pin mode

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

The possible on-chip resistor configurations are pull-up enabled, pull-down enabled, or no pull-up/pull-down. The default value is pull-up enabled.

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

6.4.3 Hysteresis

The input buffer for digital functions can be configured with hysteresis or as plain buffer through the IOCON registers.

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

6.4.4 Input inverter

If the input inverter is enabled, a HIGH pin level is inverted to 0 and a LOW pin level is inverted to 1.

6.4.5 Open-drain mode

A pseudo open-drain mode can be supported for all digital pins. Note that except for the I^2C-bus pins, this is not a true open-drain mode.

6.4.6 Analog mode

In analog mode, the digital receiver is disconnected to obtain an accurate input voltage for analog-to-digital conversions or the crystal oscillator inputs. This mode can be selected in those IOCON registers that control pins with an analog function. If analog mode is selected, hysteresis, pin mode, inverter, glitch filter, and open-drain settings have no effect. Disable the pull-up and pull-down resistors before using the analog functions.

For pins without analog functions, the analog mode setting has no effect.

6.4.7 I^2C mode

If the I^2C function is selected by the FUNC bits of registers PIO0_4 and PIO0_5 (Table 90), then the I^2C-bus pins can be configured for different I^2C-modes:

- Standard mode/Fast-mode I^2C with 50 ns input glitch filter. An open-drain output according to the I^2C-bus specification can be configured separately.
- Fast-mode Plus I^2C with 50 ns input glitch filter. In this mode, the pins function as high-current sinks. An open-drain output according to the I^2C-bus specification can be configured separately.
- Standard functionality without input filter.

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

6.4.8 $\overline{\text{RESET}}$ pin (RESET_PIO0_0)

See Figure 9 for the reset pad configuration. $\overline{\text{RESET}}$ 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. The reset pin includes a fixed 20 ns glitch filter.

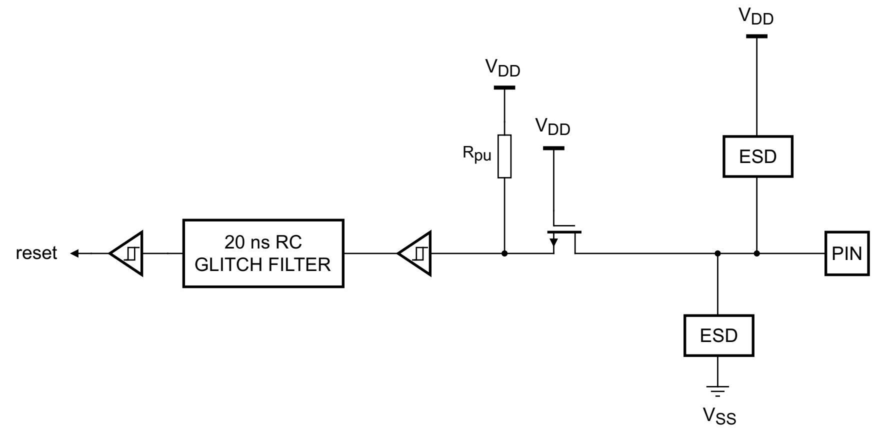


Fig 9. Reset pad configuration

6.4.9 WAKEUP pin (PIO0_16)

The WAKEUP pin is combined with pin PIO0_16 and includes a 20 ns fixed glitch filter. This pin must be pulled HIGH externally before entering Deep power-down mode and pulled LOW to exit Deep power-down mode. A LOW-going pulse as short as 50 ns wakes up the part. The WAKEUP pin is the only pin powered in Deep power-down mode.

6.4.10 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 is derived from the main clock using the IOCONCLKDIV register divided by the CLKDIV value (PCLKn). 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 57 “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

The I/O configuration registers control the PIO port pins, the inputs and outputs of all peripherals and functional blocks, the I^{2}C-bus pins, and the ADC input pins.

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

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

Name	Access	Address offset	Description	Reset value	Reference
RESET_PIO0_0	R/W	0x000	I/O configuration for pin $\overline{\text{RESET}}$/PIO0_0	0x0000 0090	Table 84
PIO0_1	R/W	0x004	I/O configuration for pin PIO0_1/CLKOUT/CT32B0_MAT2/USB_FTOGGLE	0x0000 0090	Table 84
PIO0_2	R/W	0x008	I/O configuration for pin PIO0_2/SSP0_SSEL/CT16B0_CAP0/R_0	0x0000 0090	Table 84
PIO0_3	R/W	0x00C	I/O configuration for pin PIO0_3/USB_VBUS/R_1	0x0000 0090	Table 84
PIO0_4	R/W	0x010	I/O configuration for pin PIO0_4/I2C0_SCL/R_2	0x0000 0000	Table 90
PIO0_5	R/W	0x014	I/O configuration for pin PIO0_5/I2C0_SDA/R_3	0x0000 0000	Table 90
PIO0_6	R/W	0x018	I/O configuration for pin PIO0_6/R/SSP0_SCK/R_4	0x0000 0090	Table 84
PIO0_7	R/W	0x01C	I/O configuration for pin PIO0_7/U0_nCTS/R_5/I2C1_SCL	0x0000 0090	Table 84
PIO0_8	R/W	0x020	I/O configuration for pin PIO0_8/SSP0_MISO/CT16B0_MAT0/R/R_6	0x0000 0090	Table 84
PIO0_9	R/W	0x024	I/O configuration for pin PIO0_9/SSP0_MOSI/CT16B0_MAT1/R/R_7	0x0000 0090	Table 84
SWCLK_PIO0_10	R/W	0x028	I/O configuration for pin SWCLK/PIO0_10/SSP0_SCK/CT16B0_MAT2	0x0000 0090	Table 84
TDI_PIO0_11	R/W	0x02C	I/O configuration for pin TDI/PIO0_11/ADC_9/CT32B0_MAT3/U1_nRTS/U1_SCLK	0x0000 0090	Table 87
TMS_PIO0_12	R/W	0x030	I/O configuration for pin TMS/PIO0_12/ADC_8/CT32B1_CAP0/U1_nCTS	0x0000 0090	Table 87
TDO_PIO0_13	R/W	0x034	I/O configuration for pin TDO/PIO0_13/ADC_7/CT32B1_MAT0/U1_RXD	0x0000 0090	Table 87
TRST_PIO0_14	R/W	0x038	I/O configuration for pin nTRST/PIO0_14/ADC_6/CT32B1_MAT1/U1_TXD	0x0000 0090	Table 87
SWDIO_PIO0_15	R/W	0x03C	I/O configuration for pin SWDIO/PIO0_15/ADC_3/CT32B1_MAT2	0x0000 0090	Table 87
PIO0_16	R/W	0x040	I/O configuration for pin PIO0_16/ADC_2/CT32B1_MAT3/R_8/WAKEUP	0x0000 0090	Table 87
PIO0_17	R/W	0x044	I/O configuration for pin PIO0_17/U0_nRTS/CT32B0_CAP0/U0_SCLK	0x0000 0090	Table 84
PIO0_18	R/W	0x048	I/O configuration for pin PIO0_18/U0_RXD/CT32B0_MAT0	0x0000 0090	Table 84
PIO0_19	R/W	0x04C	I/O configuration for pin PIO0_19/U0_TXD/CT32B0_MAT1	0x0000 0090	Table 84
PIO0_20	R/W	0x050	I/O configuration for pin PIO0_20/CT16B1_CAP0/U2_RXD	0x0000 0090	Table 84

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

Name	Access	Address offset	Description	Reset value	Reference
PIO0_21	R/W	0x054	I/O configuration for pin PIO0_21/CT16B1_MAT0/SSP1_MOSI	0x0000 0090	Table 84
PIO0_22	R/W	0x058	I/O configuration for pin PIO0_22/ADC_11/CT16B1_CAP1/SSP1_MISO	0x0000 0090	Table 87
PIO0_23	R/W	0x05C	I/O configuration for pin PIO0_23/ADC_1/R_9/U0_nRI/SSP1_SSEL	0x0000 0090	Table 87
PIO1_0	R/W	0x060	I/O configuration for pin PIO1_0/CT32B1_MAT0/R_10/U2_TXD	0x0000 0090	Table 85
PIO1_1	R/W	0x064	PIO1_1/CT32B1_MAT1/R_11/U0_nDTR	0x0000 0090	Table 85
PIO1_2	R/W	0x068	I/O configuration for pin PIO1_2/CT32B1_MAT2/R_12/U1_RXD	0x0000 0090	Table 85
PIO1_3	R/W	0x06C	I/O configuration for pin PIO1_3/CT32B1_MAT3/R_13/I2C1_SDA/ADC_5	0x0000 0090	Table 88
PIO1_4	R/W	0x070	I/O configuration for pin PIO1_4/CT32B1_CAP0/R_14/U0_nDSR	0x0000 0090	Table 88
PIO1_5	R/W	0x074	I/O configuration for pin PIO1_5/CT32B1_CAP1/R_15/U0_nDCD	0x0000 0090	Table 85
PIO1_6	R/W	0x078	I/O configuration for pin PIO1_6/R_16/U2_RXD/CT32B0_CAP2	0x0000 0090	Table 85
PIO1_7	R/W	0x07C	I/O configuration for pin PIO1_7/R_17/U2_nCTS/CT16B1_CAP0	0x0000 0090	Table 85
PIO1_8	R/W	0x080	I/O configuration for pin PIO1_8/R_18/U1_TXD/CT16B0_CAP0	0x0000 0090	Table 85
PIO1_9	R/W	0x084	I/O configuration for pin PIO1_9/U0_nCTS/CT16B1_MAT1/ADC_0	0x0000 0090	Table 88
PIO1_10	R/W	0x088	I/O configuration for pin PIO1_10/U2_nRTS/U2_SCLK/CT16B1_MAT0	0x0000 0090	Table 85
PIO1_11	R/W	0x08C	I/O configuration for pin PIO1_11/I2C1_SCL/CT16B0_MAT2//U0_nRI	0x0000 0090	Table 85
PIO1_12	R/W	0x090	I/O configuration for pin PIO1_12/SSP0_MOSI/CT16B0_MAT1/R_21	0x0000 0090	Table 85
PIO1_13	R/W	0x094	I/O configuration for pin PIO1_13/U1_nCTS/SCT0_OUT3/R_22	0x0000 0090	Table 85
PIO1_14	R/W	0x098	I/O configuration for pin PIO1_14/I2C1_SDA/CT32B1_MAT2/R_23	0x0000 0090	Table 85
PIO1_15	R/W	0x09C	I/O configuration for pin PIO1_15/SSP0_SSEL/CT32B1_MAT3/R_24	0x0000 0090	Table 85
PIO1_16	R/W	0x0A0	I/O configuration for pin PIO1_16/SSP0_MISO/CT16B0_MAT0/R_25	0x0000 0090	Table 85
PIO1_17	R/W	0x0A4	I/O configuration for PIO1_17/CT16B0_CAP2/U0_RXD/R_26	0x0000 0090	Table 85
PIO1_18	R/W	0x0A8	I/O configuration for PIO1_18/CT16B1_CAP1/U0_TXD/R_27	0x0000 0090	Table 85
PIO1_19	R/W	0x0AC	I/O configuration for pin PIO1_19/U2_nCTS/SCT0_OUT0/R_28	0x0000 0090	Table 85

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

Name	Access	Address offset	Description	Reset value	Reference
PIO1_20	R/W	0x0B0	I/O configuration for pin PIO1_20/U0_nDSR/SSP1_SCK/CT16B0_MAT0	0x0000 0090	Table 85
PIO1_21	R/W	0x0B4	I/O configuration for pin PIO1_21/U0_nDCD/SSP1_MISO/CT16B0_CAP2	0x0000 0090	Table 85
PIO1_22	R/W	0x0B8	I/O configuration for pin PIO1_22/SSP1_MOSI/CT32B1_CAP1/ADC_4/R_29	0x0000 0090	Table 88
PIO1_23	R/W	0x0BC	I/O configuration for pin PIO1_23/CT16B1_MAT1/SSP1_SSEL/U2_TXD	0x0000 0090	Table 85
PIO1_24	R/W	0x0C0	I/O configuration for pin PIO1_24/CT32B0_MAT0/I2C1_SDA	0x0000 0090	Table 85
PIO1_25	R/W	0x0C4	I/O configuration for pin PIO1_25/U2_nRTS/U2_SCLK/SCT0_IN0/R_30	0x0000 0090	Table 85
PIO1_26	R/W	0x0C8	I/O configuration for pin PIO1_26/CT32B0_MAT2/U0_RXD/R_19	0x0000 0090	Table 85
PIO1_27	R/W	0x0CC	I/O configuration for pin PIO1_27/CT32B0_MAT3/U0_TXD/R_20/SSP1_SCK	0x0000 0090	Table 85
PIO1_28	R/W	0x0D0	I/O configuration for pin PIO1_28/CT32B0_CAP0/U0_SCLK/U0_nRTS	0x0000 0090	Table 85
PIO1_29	R/W	0x0D4	I/O configuration for pin PIO1_29/SSP0_SCK/CT32B0_CAP2/U0_nDTR/ADC_10	0x0000 0090	Table 88
PIO1_30	R/W	0x0D8	I/O configuration for pin PIO1_30/I2C1_SCL/SCT0_IN3/R_31	0x0000 0090	Table 85
PIO1_31	R/W	0x0DC	I/O configuration for pin PIO1_31	0x0000 0090	Table 85
-	-	0xE0 - 0xEC	Reserved	-	-
PIO2_0	R/W	0x0F0	I/O configuration for pin PIO2_0/XTALIN	0x0000 0090	Table 89
PIO2_1	R/W	0x0F4	I/O configuration for pin PIO2_1/XTALOUT	0x0000 0090	Table 89
-	-	0x0F8	Reserved	-	-
PIO2_2	R/W	0x0FC	I/O configuration for pin PIO2_2/U3_nRTS/U3_SCLK/SCT0_OUT1	0x0000 0090	Table 86
PIO2_3	R/W	0x100	I/O configuration for pin PIO2_3/U3_RXD/CT32B0_MAT1	0x0000 0090	Table 86
PIO2_4	R/W	0x104	I/O configuration for pin PIO2_4/U3_TXD/CT32B0_MAT2	0x0000 0090	Table 86
PIO2_5	R/W	0x108	I/O configuration for pin PIO2_5/U3_nCTS/SCT0_IN1	0x0000 0090	Table 86
PIO2_6	R/W	0x10C	I/O configuration for pin PIO2_6/U1_nRTS/U1_SCLK/SCT0_IN2	0x0000 0090	Table 86
PIO2_7	R/W	0x110	I/O configuration for pin PIO2_7/SSP0_SCK/SCT0_OUT2	0x0000 0090	Table 86
PIO2_8	R/W	0x114	I/O configuration for pin PIO2_8/SCT1_IN0	0x0000 0090	Table 86
PIO2_9	R/W	0x118	I/O configuration for pin PIO2_9/SCT1_IN1	0x0000 0090	Table 86
PIO2_10	R/W	0x11C	I/O configuration for pin PIO2_10/U4_nRTS/U4_SCLK	0x0000 0090	Table 86
PIO2_11	R/W	0x120	I/O configuration for pin PIO2_11/U4_RXD	0x0000 0090	Table 86

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

Name	Access	Address offset	Description	Reset value	Reference
PIO2_12	R/W	0x124	I/O configuration for pin PIO2_12/U4_TXD	0x0000 0090	Table 86
PIO2_13	R/W	0x128	I/O configuration for pin PIO2_13/U4_nCTS	0x0000 0090	Table 86
PIO2_14	R/W	0x12C	I/O configuration for pin PIO2_14/SCT1_IN2	0x0000 0090	Table 86
PIO2_15	R/W	0x130	I/O configuration for pin PIO2_15/SCT1_IN3	0x0000 0090	Table 86
PIO2_16	R/W	0x134	I/O configuration for pin PIO2_16/SCT1_OUT0	0x0000 0090	Table 86
PIO2_17	R/W	0x138	I/O configuration for PIO2_17/SCT1_OUT1	0x0000 0090	Table 86
PIO2_18	R/W	0x13C	I/O configuration for PIO2_18/SCT1_OUT2	0x0000 0090	Table 86
PIO2_19	R/W	0x140	I/O configuration for pin PIO2_19/SCT1_OUT3	0x0000 0090	Table 86
PIO2_20	R/W	0x144	I/O configuration for pin PIO2_20	0x0000 0090	Table 86
PIO2_21	R/W	0x148	I/O configuration for pin PIO2_21	0x0000 0090	Table 86
PIO2_22	R/W	0x14C	I/O configuration for pin PIO2_22	0x0000 0090	Table 86
PIO2_23	R/W	0x150	I/O configuration for pin PIO2_23	0x0000 0090	Table 86

Table 82. I/O configuration register types

Name	Address offset	True open-drain	Analog	Glitch filter on/off	Digital filter	High-drive output	Reference
RESET_PIO0_0	0x000	no	no	no	yes	no	Table 84
PIO0_1	0x004	no	no	no	yes	no	Table 84
PIO0_2	0x008	no	no	no	yes	no	Table 84
PIO0_3	0x00C	no	no	no	yes	no	Table 84
PIO0_4	0x010	yes	no	no	yes	no	Table 90
PIO0_5	0x014	yes	no	no	yes	no	Table 90
PIO0_6	0x018	no	no	no	yes	no	Table 84
PIO0_7	0x01C	no	no	no	yes	yes	Table 84
PIO0_8	0x020	no	no	no	yes	no	Table 84
PIO0_9	0x024	no	no	no	yes	no	Table 84
SWCLK_PIO0_10	0x028	no	no	no	yes	no	Table 84
TDI_PIO0_11	0x02C	no	yes	yes	yes	no	Table 87
TMS_PIO0_12	0x030	no	yes	yes	yes	no	Table 87
TDO_PIO0_13	0x034	no	yes	yes	yes	no	Table 87
TRST_PIO0_14	0x038	no	yes	yes	yes	no	Table 87
SWDIO_PIO0_15	0x03C	no	yes	yes	yes	no	Table 87
PIO0_16	0x040	no	yes	yes	yes	no	Table 87
PIO0_17	0x044	no	no	no	yes	no	Table 84
PIO0_18	0x048	no	no	no	yes	no	Table 84
PIO0_19	0x04C	no	no	no	yes	no	Table 84
PIO0_20	0x050	no	no	no	yes	no	Table 84
PIO0_21	0x054	no	no	no	yes	no	Table 84
PIO0_22	0x058	no	yes	yes	yes	no	Table 87
PIO0_23	0x05C	no	yes	yes	yes	no	Table 87
PIO1_0	0x060	no	no	no	yes	no	Table 85

Table 82. I/O configuration register types

Name	Address offset	True open-drain	Analog	Glitch filter on/off	Digital filter	High-drive output	Reference
PIO1_1	0x064	no	no	no	yes	no	Table 85
PIO1_2	0x068	no	no	no	yes	no	Table 85
PIO1_3	0x06C	no	yes	yes	yes	no	Table 88
PIO1_4	0x070	no	no	no	yes	no	Table 85
PIO1_5	0x074	no	no	no	yes	no	Table 85
PIO1_6	0x078	no	no	no	yes	no	Table 85
PIO1_7	0x07C	no	no	no	yes	no	Table 85
PIO1_8	0x080	no	no	no	yes	no	Table 85
PIO1_9	0x084	no	yes	yes	yes	no	Table 88
PIO1_10	0x088	no	no	no	yes	no	Table 85
PIO1_11	0x08C	no	no	no	yes	no	Table 85
PIO1_12	0x090	no	no	no	yes	no	Table 85
PIO1_13	0x094	no	no	no	yes	no	Table 85
PIO1_14	0x098	no	no	no	yes	no	Table 85
PIO1_15	0x09C	no	no	no	yes	no	Table 85
PIO1_16	0x0A0	no	no	no	yes	no	Table 85
PIO1_17	0x0A4	no	no	no	yes	no	Table 85
PIO1_18	0x0A8	no	no	no	yes	no	Table 85
PIO1_19	0x0AC	no	no	no	yes	no	Table 85
PIO1_20	0x0B0	no	no	no	yes	no	Table 85
PIO1_21	0x0B4	no	no	no	yes	no	Table 85
PIO1_22	0x0B8	no	yes	yes	yes	no	Table 88
PIO1_23	0x0BC	no	no	no	yes	no	Table 85
PIO1_24	0x0C0	no	no	no	yes	no	Table 85
PIO1_25	0x0C4	no	no	no	yes	no	Table 85
PIO1_26	0x0C8	no	no	no	yes	no	Table 85
PIO1_27	0x0CC	no	no	no	yes	no	Table 85
PIO1_28	0x0D0	no	no	no	yes	no	Table 85
PIO1_29	0x0D4	no	yes	yes	yes	no	Table 88
PIO1_30	0x0D8	no	no	no	yes	no	Table 85
PIO1_31	0x0DC	no	no	no	yes	yes	Table 85
-	0xE0 - 0xEC	-	-	-	-	-	-
PIO2_0	0x0F0	no	yes	yes	yes	no	Table 89
PIO2_1	0x0F4	no	yes	yes	yes	no	Table 89
-	0x0F8	-	-	-	-	-	-
PIO2_2	0x0FC	no	no	no	yes	no	Table 86
PIO2_3	0x100	no	no	no	yes	no	Table 86
PIO2_4	0x104	no	no	no	yes	no	Table 86
PIO2_5	0x108	no	no	no	yes	no	Table 86

Table 82. I/O configuration register types

Name	Address offset	True open-drain	Analog	Glitch filter on/off	Digital filter	High-drive output	Reference
PIO2_6	0x10C	no	no	no	yes	no	Table 86
PIO2_7	0x110	no	no	no	yes	no	Table 86
PIO2_8	0x114	no	no	no	yes	no	Table 86
PIO2_9	0x118	no	no	no	yes	no	Table 86
PIO2_10	0x11C	no	no	no	yes	no	Table 86
PIO2_11	0x120	no	no	no	yes	no	Table 86
PIO2_12	0x124	no	no	no	yes	no	Table 86
PIO2_13	0x128	no	no	no	yes	no	Table 86
PIO2_14	0x12C	no	no	no	yes	no	Table 86
PIO2_15	0x130	no	no	no	yes	no	Table 86
PIO2_16	0x134	no	no	no	yes	no	Table 86
PIO2_17	0x138	no	no	no	yes	no	Table 86
PIO2_18	0x13C	no	no	no	yes	no	Table 86
PIO2_19	0x140	no	no	no	yes	no	Table 86
PIO2_20	0x144	no	no	no	yes	no	Table 86
PIO2_21	0x148	no	no	no	yes	no	Table 86
PIO2_22	0x14C	no	no	no	yes	no	Table 86
PIO2_23	0x150	no	no	no	yes	no	Table 86

Table 83. IOCON function assignments

Symbol	Func0	Func1	Func2	Func3	Func4	Func5	Reference
RESET/PIO0_0	RESET	PIO0_0	-	-	-		Table 84
PIO0_1	PIO0_1	CLKOUT	CT32B0_MAT2	USB_FTOGGLE	-		Table 84
PIO0_2	PIO0_2	SSP0_SSEL	CT16B0_CAP0	R_0	-		Table 84
PIO0_3	PIO0_3	USB_VBUS	R_1	-	-		Table 84
PIO0_4	PIO0_4	I2C0_SCL	R_2	-	-		Table 90
PIO0_5	PIO0_5	I2C0_SDA	R_3	-	-		Table 90
PIO0_6	PIO0_6	R	SSP0_SCK	R_4	-		Table 84
PIO0_7	PIO0_7	U0_CTS	R_5	I2C1_SCL	-		Table 84
PIO0_8	PIO0_8	SSP0_MISO	CT16B0_MAT0	R_6	-		Table 84
PIO0_9	PIO0_9	SSP0_MOSI	CT16B0_MAT1	R_7	-		Table 84
SWCLK/PIO0_10	SWCLK	PIO0_10	SSP0_SCK	CT16B0_MAT2	-		Table 84
TDI/PIO0_11	TDI	PIO0_11	ADC_9	CT32B0_MAT3	U1_RTS	U1_SCLK	Table 87
TMS/PIO0_12	TMS	PIO0_12	ADC_8	CT32B1_CAP0	U1_CTS	PIO0_12	Table 87
TDO/PIO0_13	TDO	PIO0_13	ADC_7	CT32B1_MAT0	U1_RXD	PIO0_13	Table 87
$\overline{\text{TRST}}$/PIO0_14	$\overline{\text{TRST}}$	PIO0_14	ADC_6	CT32B1_MAT1	U1_TXD	-	Table 87
SWDIO/PIO0_15	SWDIO	PIO0_15	ADC_3	CT32B1_MAT2	-		Table 87
PIO0_16/ WAKEUP	PIO0_16	ADC_2	CT32B1_MAT3	R_8	-		Table 87
PIO0_17	PIO0_17	U0_RTS	CT32B0_CAP0	U0_SCLK	-		Table 84
PIO0_18	PIO0_18	U0_RXD	CT32B0_MAT0	-	-		Table 84
PIO0_19	PIO0_19	U0_TXD	CT32B0_MAT1	-	-		Table 84
PIO0_20	PIO0_20	CT16B1_CAP0	U2_RXD	-	-		Table 84
PIO0_21	PIO0_21	CT16B1_MAT0	SSP1_MOSI	-	-		Table 84
PIO0_22	PIO0_22	ADC_11	CT16B1_CAP1	SSP1_MISO	-		Table 87
PIO0_23	PIO0_23	ADC_1	R_9	U0_RI	SSP1_SSEL		Table 87
PIO1_0	PIO1_0	CT32B1_MAT0	R_10	U2_TXD	-		Table 85
PIO1_1	PIO1_1	CT32B1_MAT1	R_11	U0_DTR	-		Table 85
PIO1_2	PIO1_2	CT32B1_MAT2	R_12	U1_RXD	-		Table 85
PIO1_3	PIO1_3	CT32B1_MAT3	R_13	I2C1_SDA	ADC_5		Table 88
PIO1_4	PIO1_4	CT32B1_CAP0	R_14	U0_DSR	-		Table 85
PIO1_5	PIO1_5	CT32B1_CAP1	R_15	U0_DCD	-		Table 85

Table 83. IOCON function assignments

Symbol	Func0	Func1	Func2	Func3	Func4	Func5	Reference
PIO1_6	PIO1_6	R_16	U2_RXD	CT32B0_CAP2	-		Table 85
PIO1_7	PIO1_7	R_17	U2_CTS	CT16B1_CAP0	-		Table 85
PIO1_8	PIO1_8	R_18	U1_TXD	CT16B0_CAP0	-		Table 85
PIO1_9	PIO1_9	U0_CTS	CT16B1_MAT1	ADC_0	-		Table 88
PIO1_10	PIO1_10	U2_RTS	U2_SCLK	CT16B1_MAT0	-		Table 85
PIO1_11	PIO1_11	I2C1_SCL	CT16B0_MAT2	U0_RI	-		Table 85
PIO1_12	PIO1_12	SSP0_MOSI	CT16B0_MAT1	R_21	-		Table 85
PIO1_13	PIO1_13	U1_CTS	SCT0_OUT3	R_22	-		Table 85
PIO1_14	PIO1_14	I2C1_SDA	CT32B1_MAT2	R_23	-		Table 85
PIO1_15	PIO1_15	SSP0_SSEL	CT32B1_MAT3	R_24	-		Table 85
PIO1_16	PIO1_16	SSP0_MISO	CT16B0_MAT0	R_25	-		Table 85
PIO1_17	PIO1_17	CT16B0_CAP2	U0_RXD	R_26	-		Table 85
PIO1_18	PIO1_18	CT16B1_CAP1	U0_TXD	R_27	-		Table 85
PIO1_19	PIO1_19	U2_CTS	SCT0_OUT0	R_28	-		Table 85
PIO1_20	PIO1_20	U0_DSR	SSP1_SCK	CT16B0_MAT0	-		Table 85
PIO1_21	PIO1_21	U0_DCD	SSP1_MISO	CT16B0_CAP2	-		Table 85
PIO1_22	PIO1_22	SSP1_MOSI	CT32B1_CAP1	ADC_4	R_29		Table 88
PIO1_23	PIO1_23	CT16B1_MAT1	SSP1_SSEL	U2_TXD	-		Table 85
PIO1_24	PIO1_24	CT32B0_MAT0	I2C1_SDA	-	-		Table 85
PIO1_25	PIO1_25	U2_RTS	U2_SCLK	SCT0_IN0	R_30		Table 85
PIO1_26	PIO1_26	CT32B0_MAT2	U0_RXD	R_19			Table 85
PIO1_27	PIO1_27	CT32B0_MAT3	U0_TXD	R_20	SSP1_SCK		Table 85
PIO1_28	PIO1_28	CT32B0_CAP0	U0_SCLK	U0_RTS	-		Table 85
PIO1_29	PIO1_29	SSP0_SCK	CT32B0_CAP2	U0_DTRn	ADC_10		Table 88
PIO1_30	PIO1_30	I2C1_SCL	SCT0_IN3	R_31	-		Table 85
PIO1_31	PIO1_31	-	-	-	-		Table 85
PIO2_0	PIO2_0	XTALIN	-	-	-		Table 89
PIO2_1	PIO2_1	XTALOUT	-	-	-		Table 89
PIO2_2	PIO2_2	U3_RTS	U3_SCLK	SCT0_OUT1	-		Table 86
PIO2_3	PIO2_3	U3_RXD	CT32B0_MAT1	-	-		Table 86
PIO2_4	PIO2_4	U3_TXD	CT32B0_MAT2	-	-		Table 86

Table 83. IOCON function assignments

Symbol	Func0	Func1	Func2	Func3	Func4	Func5	Reference
PIO2_5	PIO2_5	U3_CTS	SCT0_IN1	-	-		Table 86
PIO2_6	PIO2_6	U1_RTS	U1_SCLK	SCT0_IN2			Table 86
PIO2_7	PIO2_7	SSP0_SCK	SCT0_OUT2	-	-		Table 86
PIO2_8	PIO2_8	SCT1_IN0	-	-	-		Table 86
PIO2_9	PIO2_9	SCT1_IN1	-	-	-		Table 86
PIO2_10	PIO2_10	U4_RTS	U4_SCLK	-	-		Table 86
PIO2_11	PIO2_11	U4_RXD	-	-	-		Table 86
PIO2_12	PIO2_12	U4_TXD	-	-	-		Table 86
PIO2_13	PIO2_13	U4_CTS	-	-	-		Table 86
PIO2_14	PIO2_14	SCT1_IN2	-	-	-		Table 86
PIO2_15	PIO2_15	SCT1_IN3	-	-	-		Table 86
PIO2_16	PIO2_16	SCT1_OUT0	-	-	-		Table 86
PIO2_17	PIO2_17	SCT1_OUT1	-	-	-		Table 86
PIO2_18	PIO2_18	SCT1_OUT2	-	-	-		Table 86
PIO2_19	PIO2_19	SCT1_OUT3	-	-	-		Table 86
PIO2_20	PIO2_20	-	-	-	-		Table 86
PIO2_21	PIO2_21	-	-	-	-		Table 86
PIO2_22	PIO2_22	-	-	-	-		Table 86
PIO2_23	PIO2_23	-	-	-	-		Table 86

6.5.1 Pin control registers for standard digital I/O pins

These registers control the standard digital I/O pins without analog pads on each port (including the RESET pin). The programmable glitch filter clock frequencies are configured in the SYSCON block (see Table 57). For the glitch filter time constant, select one of the IOCON divider clocks.

Table 84. Digital pin control registers (PIO0_[0:3], addresses 0x4004 4000 (PIO0_0) to 0x4004 400C (PIO0_3); PIO0_[6:10], addresses 0x4004 4016 (PIO0_6) to 0x4004 4028 (PIO0_10); PIO0_[17:21], addresses 0x4004 4044 (PIO0_17) to 0x4004 4054 (PIO0_21)) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	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	Enable. 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.	

Table 84. Digital pin control registers (PIO0_[0:3], addresses 0x4004 4000 (PIO0_0) to 0x4004 400C (PIO0_3); PIO0_[6:10], addresses 0x4004 4016 (PIO0_6) to 0x4004 4028 (PIO0_10); PIO0_[17:21], addresses 0x4004 4044 (PIO0_17) to 0x4004 4054 (PIO0_21)) bit description

Bit	Symbol	Value	Description	Reset value
15:13	CLKDIV		Select peripheral clock divider for input filter sampling clock IOCONCLKDIV. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0. Use IOCON clock divider 0.	
		0x1	IOCONCLKDIV1. Use IOCON clock divider 1.	
		0x2	IOCONCLKDIV2. Use IOCON clock divider 2.	
		0x3	IOCONCLKDIV3. Use IOCON clock divider 3.	
		0x4	IOCONCLKDIV4. Use IOCON clock divider 4.	
		0x5	IOCONCLKDIV5. Use IOCON clock divider 5.	
		0x6	IOCONCLKDIV6. Use IOCON clock divider 6.	
31:16	-	-	Reserved.	0

Table 85. Digital pin control registers (PIO1_[0:2], addresses 0x4004 4060 (PIO1_0) to 0x4004 4068 (PIO1_2); PIO1_[4:8], addresses 0x4004 4070 (PIO1_4) to 0x4004 4080 (PIO1_8); PIO1_[10:21], addresses 0x4004 4088 (PIO0_10) to 0x4004 40B4 (PIO1_21); PIO1_[23:28], addresses 0x4004 40BC (PIO1_23) to 0x4004 40D0 (PIO1_28); PIO1_[30:31], addresses 0x4004 40D8 (PIO1_30) to 0x4004 40DC (PIO1_31)) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	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	Enabled. Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	

Table 85. Digital pin control registers (PIO1_[0:2], addresses 0x4004 4060 (PIO1_0) to 0x4004 4068 (PIO1_2); PIO1_[4:8], addresses 0x4004 4070 (PIO1_4) to 0x4004 4080 (PIO1_8); PIO1_[10:21], addresses 0x4004 4088 (PIO0_10) to 0x4004 40B4 (PIO1_21); PIO1_[23:28], addresses 0x4004 40BC (PIO1_23) to 0x4004 40D0 (PIO1_28); PIO1_[30:31], addresses 0x4004 40D8 (PIO1_30) to 0x4004 40DC (PIO1_31)) bit description

Bit	Symbol	Value	Description	Reset value
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLKDIV		Select peripheral clock divider for input filter sampling clock IOCONCLKDIV. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0. Use IOCON clock divider 0.	
		0x1	IOCONCLKDIV1. Use IOCON clock divider 1.	
		0x2	IOCONCLKDIV2. Use IOCON clock divider 2.	
		0x3	IOCONCLKDIV3. Use IOCON clock divider 3.	
		0x4	IOCONCLKDIV4. Use IOCON clock divider 4.	
		0x5	IOCONCLKDIV5. Use IOCON clock divider 5.	
		0x6	IOCONCLKDIV6. Use IOCON clock divider 6.	
31:16	-	-	Reserved.	0

Table 86. Digital pin control registers (PIO2_[2:23], addresses 0x4004 40FC(PIO2_2) to 0x4004 414C (PIO2_23)) bit description

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

Table 86. Digital pin control registers (PIO2_[2:23], addresses 0x4004 40FC(PIO2_2) to 0x4004 414C (PIO2_23)) bit description

Bit	Symbol	Value	Description	Reset value
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Enabled. 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	CLKDIV		Select peripheral clock divider for input filter sampling clock IOCONCLKDIV. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0. Use IOCON clock divider 0.	
		0x1	IOCONCLKDIV1. Use IOCON clock divider 1.	
		0x2	IOCONCLKDIV2. Use IOCON clock divider 2.	
		0x3	IOCONCLKDIV3. Use IOCON clock divider 3.	
		0x4	IOCONCLKDIV4. Use IOCON clock divider 4.	
		0x5	IOCONCLKDIV5. Use IOCON clock divider 5.	
		0x6	IOCONCLKDIV6. Use IOCON clock divider 6.	
31:16	-	-	Reserved.	0

6.5.2 Pin control registers for digital/analog I/O pins

These registers control the digital I/O pins with analog pads (ADC inputs and the XTALIN and XTALOUT pins. The programmable glitch filter clock frequencies are configured in the SYSCON block (see Table 57). For the glitch filter time constant, select one of the IOCON divider clocks.

Table 87. Digital/analog pin control registers (PIO0_[11:16], addresses 0x4004 402C (PIO0_11) to 0x4004 4040 (PIO0_16); PIO0_[22:23], addresses 0x4004 4058 (PIO0_22) to 0x4004 405C (PIO0_23)) bit description

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

Table 87. Digital/analog pin control registers (PIO0_[11:16], addresses 0x4004 402C (PIO0_11) to 0x4004 4040 (PIO0_16); PIO0_[22:23], addresses 0x4004 4058 (PIO0_22) to 0x4004 405C (PIO0_23)) bit description

Bit	Symbol	Value	Description	Reset value
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	INV		Invert input	0
		0	Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0).	
		1	Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1).	
7	ADMODE		Analog mode.	1
		0	Enable. Pin is configured as analog input.	
		1	Disable. Pin is configured as digital I/O.	
8	FILTR		Selects fixed 10 ns input glitch filter.	0
		0	Enabled. Filter enabled.	
		1	Disabled. Filter disabled.	
9	-	-	Reserved.	0
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Enable. 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	CLKDIV		Select peripheral clock divider for input filter sampling clock IOCONCLKDIV. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0. Use IOCON clock divider 0.	
		0x1	IOCONCLKDIV1. Use IOCON clock divider 1.	
		0x2	IOCONCLKDIV2. Use IOCON clock divider 2.	
		0x3	IOCONCLKDIV3. Use IOCON clock divider 3.	
		0x4	IOCONCLKDIV4. Use IOCON clock divider 4.	
		0x5	IOCONCLKDIV5. Use IOCON clock divider 5.	
		0x6	IOCONCLKDIV6. Use IOCON clock divider 6.	
31:16	-	-	Reserved.	0

Table 88. Digital/analog pin control registers (PIO1_3, addresses 0x4004 406C; PIO1_9 address 0x4004 4084; PIO1_22, address 0x4004 40B8; PIO1_29, address 0x4004 40D4) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	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).	
7	ADMODE		Analog mode.	1
		0	Enable. Pin is configured as analog input.	
		1	Disable. Pin is configured as digital I/O.	
8	FILTR		Selects fixed 10 ns input glitch filter.	0
		0	Enabled. Filter enabled.	
		1	Disabled. Filter disabled.	
9	-	-	Reserved.	0
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Enabled. 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.	

Table 88. Digital/analog pin control registers (PIO1_3, addresses 0x4004 406C; PIO1_9 address 0x4004 4084; PIO1_22, address 0x4004 40B8; PIO1_29, address 0x4004 40D4) bit description

Bit	Symbol	Value	Description	Reset value
15:13	CLKDIV		Select peripheral clock divider for input filter sampling clock IOCONCLKDIV. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0. Use IOCON clock divider 0.	
		0x1	IOCONCLKDIV1. Use IOCON clock divider 1.	
		0x2	IOCONCLKDIV2. Use IOCON clock divider 2.	
		0x3	IOCONCLKDIV3. Use IOCON clock divider 3.	
		0x4	IOCONCLKDIV4. Use IOCON clock divider 4.	
		0x5	IOCONCLKDIV5. Use IOCON clock divider 5.	
		0x6	IOCONCLKDIV6. Use IOCON clock divider 6.	
31:16	-	-	Reserved.	0

Table 89. Digital/analog pin control registers (PIO2_[0:1], addresses 0x4004 40F0 (PIO2_0) to 0x4004 40F4 (PIO2_1)) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	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).	
7	ADMODE		Analog mode.	1
		0	Enable. Pin is configured as analog input.	
		1	Disable. Pin is configured as digital I/O.	
8	FILTR		Selects fixed 10 ns input glitch filter.	0
		0	Enabled. Filter enabled.	
		1	Disabled. Filter disabled.	
9	-	-	Reserved.	0

Table 89. Digital/analog pin control registers (PIO2_[0:1], addresses 0x4004 40F0 (PIO2_0) to 0x4004 40F4 (PIO2_1)) bit description

Bit	Symbol	Value	Description	Reset value
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Enabled. 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	CLKDIV		Select peripheral clock divider for input filter sampling clock IOCONCLKDIV. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0. Use IOCON clock divider 0.	
		0x1	IOCONCLKDIV1. Use IOCON clock divider 1.	
		0x2	IOCONCLKDIV2. Use IOCON clock divider 2.	
		0x3	IOCONCLKDIV3. Use IOCON clock divider 3.	
		0x4	IOCONCLKDIV4. Use IOCON clock divider 4.	
		0x5	IOCONCLKDIV5. Use IOCON clock divider 5.	
		0x6	IOCONCLKDIV6. Use IOCON clock divider 6.	
31:16	-	-	Reserved.	0

6.5.3 Pin control registers for open-drain I/O pins

These registers control the digital I/O pins with true open-drain I2C pads. The programmable glitch filter clock frequencies are configured in the SYSCON block (see Table 57). For the glitch filter time constant, select one of the IOCON divider clocks.

Table 90. I2C open-drain pin control registers (PIO0_[4:5], addresses 0x4004 4010 (PIO0_4) to 0x4004 4014 (PIO0_5)) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
7:3	-	-	Reserved.	0
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 GPIO functionality. Requires external pull-up for GPIO output function.	
		0x2	Fast-mode Plus I2C	
		0x3	Reserved.	

Table 90. I2C open-drain pin control registers (PIO0_[4:5], addresses 0x4004 4010 (PIO0_4) to 0x4004 4014 (PIO0_5)) bit description

Bit	Symbol	Value	Description	Reset value
10	-	-	Reserved.	0
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	CLKDIV		Select peripheral clock divider for input filter sampling clock IOCONCLKDIV. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0. Use IOCON clock divider 0.	
		0x1	IOCONCLKDIV1. Use IOCON clock divider 1.	
		0x2	IOCONCLKDIV2. Use IOCON clock divider 2.	
		0x3	IOCONCLKDIV3. Use IOCON clock divider 3.	
		0x4	IOCONCLKDIV4. Use IOCON clock divider 4.	
		0x5	IOCONCLKDIV5. Use IOCON clock divider 5.	
		0x6	IOCONCLKDIV6. Use IOCON clock divider 6.	
31:16	-	-	Reserved.	0

UM10732

Chapter 7: LPC11U6x/E6x General-Purpose I/O (GPIO)

Rev. 1.8 — 19 December 2016 **User manual**

7.1 How to read this chapter

All GPIO registers refer to 32 pins on each port. Depending on the package type, not all pins are available, and the corresponding bits in the GPIO registers are reserved (see Table 91).

Table 91. GPIO pins available

Package	GPIO Port 0	GPIO Port 1	GPIO Port 2
LPC11U6x			
LQFP48	PIO0_0 to PIO0_23	PIO1_13, PIO1_20, PIO1_21, PIO1_23, PIO1_24	PIO2_0 to PIO2_2, PIO2_5, PIO2_7
LQFP64	PIO0_0 to PIO0_23	PIO1_0, PIO1_7, PIO1_9, PIO1_10, PIO1_13, PIO1_19 to PIO1_21, PIO1_23, PIO1_24, PIO1_26 to PIO1_30	PIO2_0 to PIO2_2, PIO2_5 to PIO2_7, PIO2_15, PIO2_18, PIO2_19
LQFP100	PIO0_0 to PIO0_23	PIO1_0 to PIO1_31	PIO2_0 to PIO2_23
LPC11E6x			
LQFP48	PIO0_0 to PIO0_23	PIO1_13, PIO1_20, PIO1_21, PIO1_23, PIO1_24	PIO2_0 to PIO2_2, PIO2_3, PIO2_4, PIO2_5, PIO2_7
LQFP64	PIO0_0 to PIO0_23	PIO1_0, PIO1_7, PIO1_9, PIO1_10, PIO1_13, PIO1_19 to PIO1_21, PIO1_23, PIO1_24, PIO1_26 to PIO1_30	PIO2_0 to PIO2_2, PIO2_3, PIO2_4, PIO2_5 to PIO2_7, PIO2_15, PIO2_18, PIO2_19
LQFP100	PIO0_0 to PIO0_23	PIO1_0 to PIO1_31	PIO2_0 to PIO2_23

7.2 Basic configuration

For the GPIO port registers, enable the clock to the GPIO port in the SYSAHBCLKCTRL register (Table 40).

7.3 Features

- GPIO pins can be configured as input or output by software.
- All GPIO pins default to inputs with interrupt disabled at reset.
- Pin registers allow pins to be sensed and set individually.

7.4 General description

The GPIO pins can be used in several ways to set pins as inputs or outputs and use the inputs as combinations of level and edge sensitive interrupts.

The GPIOs can be used as external interrupts together with the pin interrupt and group interrupt blocks, see Table 106 and Table 102.

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

7.5 Register description

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

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

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

Table 92. Register overview: GPIO port (base address 0xA000 0000)

Name	Access	Address offset	Description	Reset value	Width	Reference
B0 to B23	R/W	0x0000 to 0x0017	Byte pin registers port 0; pins PIO0_0 to PIO0_24	ext	byte (8 bit)	Table 93
-	-	0x0018 to 0x001F	Reserved	-	-	-
B32 to B63	R/W	0x0020 to 0x003F	Byte pin registers port 1	ext	byte (8 bit)	Table 93
B64 to B87	R/W	0x0040 to 0x0057	Byte pin registers port 2	ext	byte (8 bit)	Table 93
W0 to W23	R/W	0x1000 to 0x105C	Word pin registers port 0	ext	word (32 bit)	Table 94
-		0x1060 to 0x107C	Reserved	-	-	-
W32 to W63	R/W	0x1080 to 0x10FC	Word pin registers port 1	ext	word (32 bit)	Table 94
W64 to W87	R/W	0x1100 to 0x115C	Word pin registers port 2	ext	word (32 bit)	Table 94
DIR0	R/W	0x2000	Direction registers port 0	0	word (32 bit)	Table 95
DIR1	R/W	0x2004	Direction registers port 1	0	word (32 bit)	Table 95
DIR2	R/W	0x2008	Direction registers port 2	0	word (32 bit)	Table 95
MASK0	R/W	0x2080	Mask register port 0	0	word (32 bit)	Table 96
MASK1	R/W	0x2084	Mask register port 1	0	word (32 bit)	Table 96
MASK2	R/W	0x2088	Mask register port 2	0	word (32 bit)	Table 96
PIN0	R/W	0x2100	Port pin register port 0	ext	word (32 bit)	Table 97
PIN1	R/W	0x2104	Port pin register port 1	ext	word (32 bit)	Table 97
PIN2	R/W	0x2108	Port pin register port 2	ext	word (32 bit)	Table 97
MPIN0	R/W	0x2180	Masked port register port 0	ext	word (32 bit)	Table 98
MPIN1	R/W	0x2184	Masked port register port 1	ext	word (32 bit)	Table 98
MPIN2	R/W	0x2188	Masked port register port 2	ext	word (32 bit)	Table 98
SET0	R/W	0x2200	Write: Set register for port 0 Read: output bits for port 0	0	word (32 bit)	Table 99
SET1	R/W	0x2204	Write: Set register for port 1 Read: output bits for port 1	0	word (32 bit)	Table 99
SET2	R/W	0x2208	Write: Set register for port 2 Read: output bits for port 2	0	word (32 bit)	Table 99
CLR0	WO	0x2280	Clear port 0	NA	word (32 bit)	Table 100

Table 92. Register overview: GPIO port (base address 0xA000 0000)

Name	Access	Address offset	Description	Reset value	Width	Reference
CLR1	WO	0x2284	Clear port 1	NA	word (32 bit)	Table 100
CLR2	WO	0x2288	Clear port 2	NA	word (32 bit)	Table 100
NOT0	WO	0x2300	Toggle port 0	NA	word (32 bit)	Table 101
NOT1	WO	0x2304	Toggle port 1	NA	word (32 bit)	Table 101
NOT2	WO	0x2308	Toggle port 2	NA	word (32 bit)	Table 101

7.5.1 GPIO port byte pin registers

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

Remark: Registers B24 to B31 are reserved.

Table 93. GPIO port byte pin registers (B[0:B87], addresses 0xA000 0000 (B0) to 0xA000 0057 (B87)) bit description

Bit	Symbol	Description	Reset value	Access
0	PBYTE	Read: state of the pin PIOm_n, regardless of direction, masking, or alternate function, except that pins configured as analog I/O always read as 0. One register for each port pin: m = port 0 to 2; n = pin 0 to 23 for ports 0 and 2 and pin 0 to 31 for port 1. Write: loads the pin's output bit.	ext	R/W
7:1		Reserved (0 on read, ignored on write)	0	-

7.5.2 GPIO port word pin registers

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

Table 94. GPIO port word pin registers (W[0:87], addresses 0xA000 1000 (W0) to 0xA000 115C (W87)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	PWORD	Read 0: pin PIOm_n is LOW. Write 0: clear output bit. Read 0xFFFF FFFF: pin PIOm_n is HIGH. Write any value 0x0000 0001 to 0xFFFF FFFF: set output bit. **Remark:** Only 0 or 0xFFFF FFFF can be read. Writing any value other than 0 will set the output bit. One register for each port pin: m = port 0 to 2; n = pin 0 to 23 for ports 0 and 2 and pin 0 to 31 for port 1.	ext	R/W

7.5.3 GPIO port direction registers

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

Table 95. GPIO direction port register (DIR[0:2], address 0xA000 2000 (DIR0) to 0xA000 2008 (DIR2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	DIRP	Selects pin direction for pin PIOm_n (bit 0 = PIOm_0, bit 1 = PIOm_1, ..., bit 31 = PIOm_31). m = port 0 to 2; n = pin 0 to 23 for ports 0 and 2 and pin 0 to 31 for port 1. 0 = input. 1 = output.	0	R/W

7.5.4 GPIO port mask registers

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

Table 96. GPIO mask port register (MASK[0:2], address 0xA000 2080 (MASK0) to 0xA000 2088 (MASK2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	MASKP	Controls which bits corresponding to PIOm_n are active in the MPORT register (bit 0 = PIOm_0, bit 1 = PIOm_1, ..., bit 31 = PIOm_31). m = port 0 to 2; n = pin 0 to 23 for ports 0 and 2 and pin 0 to 31 for port 1. 0 = Read MPORT: pin state; write MPORT: load output bit. 1 = Read MPORT: 0; write MPORT: output bit not affected.	0	R/W

7.5.5 GPIO port pin registers

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

Table 97. GPIO port pin register (PIN[0:2], address 0xA000 2100 (PIN0) to 0xA000 2108 (PIN2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	PORT	Reads pin states or loads output bits (bit 0 = PIOm_0, bit 1 = PIOm_1, ..., bit 31 = PIOm_31). m = port 0 to 2; n = pin 0 to 23 for ports 0 and 2 and pin 0 to 31 for port 1. 0 = Read: pin is low; write: clear output bit. 1 = Read: pin is high; write: set output bit.	ext	R/W

7.5.6 GPIO masked port pin registers

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

Table 98. GPIO masked port pin register (MPIN[0:2], address 0xA000 2180 (MPIN0) to 0xA000 2188 (MPIN2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	MPORTP	Masked port register (bit 0 = PIOm_0, bit 1 =PIOm_1, ..., bit 31 = PIOm_31). m = port 0 to 2; n = pin 0 to 23 for ports 0 and 2 and pin 0 to 31 for port 1. 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

7.5.7 GPIO port set registers

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

Table 99. GPIO set port register (SET[0:2], address 0xA000 2200 (SET0) to 0xA000 2208 (SET2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	SETP	Read or set output bits (bit 0 = PIOm_0, bit 1 =PIOm_1, ..., bit 31 = PIOm_31). m = port 0 to 2; n = pin 0 to 23 for ports 0 and 2 and pin 0 to 31 for port 1. 0 = Read: output bit: write: no operation. 1 = Read: output bit; write: set output bit.	0	R/W

7.5.8 GPIO port clear registers

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

Table 100. GPIO clear port register (CLR[0:2], 0xA000 2280 (CLR0) to 0xA000 2288 (CLR2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	CLRP	Clear output bits (bit 0 = PIOm_0, bit 1 =PIOm_1, ..., bit 31 = PIOm_31). m = port 0 to 2; n = pin 0 to 23 for ports 0 and 2 and pin 0 to 31 for port 1. 0 = No operation. 1 = Clear output bit.	NA	WO

7.5.9 GPIO port toggle registers

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

Table 101. GPIO toggle port register (NOT[0:2], address 0xA000 2300 (NOT0) to 0xA000 2308 (NOT2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	NOTP	Toggle output bits (bit 0 = PIOm_0, bit 1 =PIOm_1, ..., bit 31 = PIOm_31). m = port 0 to 2; n = pin 0 to 23 for ports 0 and 2 and pin 0 to 31 for port 1. 0 = no operation. 1 = Toggle output bit.	NA	WO

7.6 Functional description

7.6.1 Reading pin state

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

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

7.6.2 GPIO output

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

1. The pin must be selected for GPIO operation in the IOCON block (this is the default), and
2. the pin must be selected for output by a 1 in its port’s DIR register.

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

There are seven 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 7.6.1 returns the state of pins, regardless of their direction or alternate functions.

7.6.3 Masked I/O

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

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

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

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

7.6.4 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.

UM10732

Chapter 8: LPC11U6x/E6x Grouped GPIO input interrupt (GINT0/1)

Rev. 1.8 — 19 December 2016 **User manual**

8.1 How to read this chapter

The grouped interrupt feature is available on all parts.

8.2 Features

- The inputs from any number of digital pins can be enabled to contribute to a combined group interrupt.
- The polarity of each input enabled for the group interrupt can be configured HIGH or LOW.
- Enabled interrupts can be logically combined through an OR or AND operation.
- Two group interrupts are supported to reflect two distinct interrupt patterns.
- The grouped interrupts can wake up the part from sleep, deep-sleep or power-down modes.

8.3 Basic configuration

For the group interrupt feature, enable the clock to both the GROUP0 and GROUP1 register interfaces in the SYSAHBCLKCTRL register ((Table 40, bits 23 and 24). The group interrupt wake-up feature is enabled in the STARTERP1 register (Table 66).

8.4 General description

The GPIO pins can be used in several ways to set pins as inputs or outputs and use the inputs as combinations of level and edge sensitive interrupts.

For each port/pin connected to one of the two the GPIO Grouped Interrupt blocks (GROUP0 and GROUP1), the GPIO grouped interrupt registers determine which pins are enabled to generate interrupts and what the active polarities of each of those inputs are.

The GPIO grouped interrupt registers also select whether the interrupt output will be level or edge triggered and whether it will be based on the OR or the AND of all of the enabled inputs.

When the designated pattern is detected on the selected input pins, the GPIO grouped interrupt block generates an interrupt. If the part is in a power-savings mode, it first asynchronously wakes the part up prior to asserting the interrupt request. The interrupt request line can be cleared by writing a one to the interrupt status bit in the control register.

8.5 Register description

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

Table 102. Register overview: GROUP0 interrupt (base address 0x4005 C000 (GINT0) and 0x4006 0000 (GINT1))

Name	Access	Address offset	Description	Reset value	Reference
CTRL	R/W	0x000	GPIO grouped interrupt control register	0	Table 103
PORT_POL0	R/W	0x020	GPIO grouped interrupt port 0 polarity register	0xFFFF FFFF	Table 104
PORT_POL1	R/W	0x024	GPIO grouped interrupt port 1 polarity register	0xFFFF FFFF	Table 104
PORT_POL2	R/W	0x028	GPIO grouped interrupt port 2 polarity register	0xFFFF FFFF	Table 104
PORT_ENA0	R/W	0x040	GPIO grouped interrupt port 0 enable register	0	Table 105
PORT_ENA1	R/W	0x044	GPIO grouped interrupt port 1 enable register	0	Table 105
PORT_ENA2	R/W	0x048	GPIO grouped interrupt port 2 enable register	0	Table 105

8.5.1 Grouped interrupt control register

Table 103. GPIO grouped interrupt control register (CTRL, addresses 0x4005 C000 (GINT0) and 0x4006 0000 (GINT1)) bit description

Bit	Symbol	Value	Description	Reset value
0	INT		Group interrupt status. This bit is cleared by writing a one to it. Writing zero has no effect.	0
		0	No interrupt request is pending.	
		1	Interrupt request is active.	
1	COMB		Combine enabled inputs for group interrupt	0
		0	OR functionality: A grouped interrupt is generated when any one of the enabled inputs is active (based on its programmed polarity).	
		1	AND functionality: An interrupt is generated when all enabled bits are active (based on their programmed polarity).	
2	TRIG		Group interrupt trigger	0
		0	Edge-triggered	
		1	Level-triggered	
31:3	-	-	Reserved	0

8.5.2 GPIO grouped interrupt port polarity registers

The grouped interrupt port polarity registers determine how the polarity of each enabled pin contributes to the grouped interrupt. Each port is associated with its own port polarity register, and the values of both registers together determine the grouped interrupt.

Each register PORT_POLm controls the polarity of pins in port m.

Table 104. GPIO grouped interrupt port polarity registers (PORT_POL[0:2], addresses 0x4005 C020 (PORT_POL0) to 0x4005 C028 (PORT_POL2) (GINT0) and 0x4006 0020 (PORT_POL0) to 0x4006 0028 (PORT_POL2) (GINT1)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	POL	Configure pin polarity of port m pins for group interrupt. Bit n corresponds to pin PIOm_n of port m. m = port 0 to 2; n = pin 0 to 23 for ports 0 and 2 and pin 0 to 31 for port 1. 0 = the pin is active LOW. If the level on this pin is LOW, the pin contributes to the group interrupt. 1 = the pin is active HIGH. If the level on this pin is HIGH, the pin contributes to the group interrupt.	1	-

8.5.3 GPIO grouped interrupt port enable registers

The grouped interrupt port enable registers enable the pins which contribute to the grouped interrupt. Each port is associated with its own port enable register, and the values of both registers together determine which pins contribute to the grouped interrupt.

Each register PORT_ENm enables pins in port m.

Table 105. GPIO grouped interrupt port enable registers (PORT_ENA[0:2], addresses 0x4005 C040 (PORT_ENA0) to 0x4005 C048 (PORT_ENA2) (GINT0) and 0x4006 0040 (PORT_ENA0) to 0x4006 0048 (PORT_ENA2) (GINT1)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	ENA	Enable port 0 pin for group interrupt. Bit n corresponds to pin Pm_n of port m. m = port 0 to 2; n = pin 0 to 23 for ports 0 and 2 and pin 0 to 31 for port 1. 0 = the port 0 pin is disabled and does not contribute to the grouped interrupt. 1 = the port 0 pin is enabled and contributes to the grouped interrupt.	0	-

8.6 Functional description

With group interrupts, any subset of the pins in each port can be selected to contribute to a common interrupt. Any of the pin and port interrupts can be enabled to wake the part from Deep-sleep mode or Power-down mode.

In this interrupt facility, an interrupt can be requested for each port, based on any selected subset of pins within each port. The pins that contribute to each port interrupt are selected by 1s in the port’s Enable register, and an interrupt polarity can be selected for each pin in the port’s Polarity register. The level on each pin is exclusive-ORed with its polarity bit and the result is ANDed with its enable bit, and these results are then inclusive-ORed among all the pins in the port, to create the port’s raw interrupt request.

The raw interrupt request from each of the two group interrupts is sent to the NVIC, which can be programmed to treat it as level- or edge-sensitive, or it can be edge-detected by the wake-up interrupt logic (see Table 6).

UM10732

Chapter 9: LPC11U6x/E6x Pin interrupt and pattern match engine (PINT)

Rev. 1.8 — 19 December 2016 **User manual**

9.1 How to read this chapter

The pin interrupt and pattern match engine is available on all parts.

9.2 Features

- Pin interrupts
  - Up to eight pins can be selected from all GPIO pins on ports 0 and 1 and from pins PIO2_0 to PIO2_7 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 8 pins can be selected from all digital pins on ports 0 and 1 and from pins PIO2_0 to PIO2_7 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.
  - Pattern match can be used, in conjunction with software, to create complex state machines based on pin inputs.

9.3 Basic configuration

- Pin interrupts:
  - Select up to eight external interrupt pins from all digital port pins on ports 0 and 1 and from pins PIO2_0 to PIO2_7 in the PINMUX block (Table 62). 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 40, bit 19).
  - 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 65).
  - Each selected pin interrupt is assigned to one interrupt in the NVIC (interrupts #0 to #7 for pin interrupts 0 to 7).
- Pattern match engine:

- Select up to eight external pins from all digital port pins on ports 0 and 1 and from pins PIO2_0 to PIO2_7 in the PINMUX block (Table 62). 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 40, bit 19).
- Each bit slice of the pattern match engine is assigned to one interrupt in the NVIC (interrupts #0 to #7 for pin interrupts 0 to 7).

9.3.1 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. 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 from ports 0 and 1and from pins PIO2_0 to PIO2_7 into one of the eight PINTSEL registers in the PINMUX block.
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 4.4.41 “Pin Interrupt Select registers 0 to 7” in the PINMUX block for the PINTSEL registers.

9.4 Pin description

The inputs to the pin interrupt and pattern match engine are determined by the pin interrupt select registers in the PINMUX. See Section 4.4.41 “Pin Interrupt Select registers 0 to 7”.

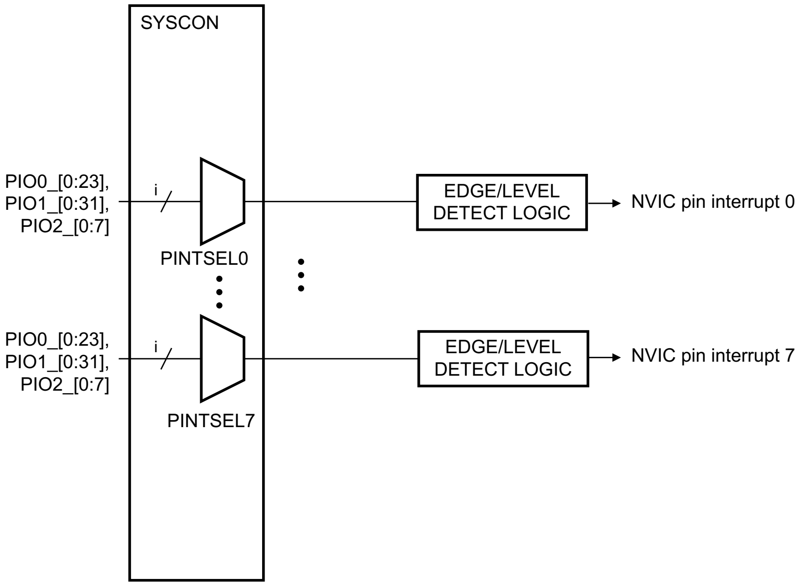
The following pins are available for the pin interrupt/pattern match engine: PIO0_0 to PIO0_23, PIO1_0 to PIO1_31, and PIO2_0 to PIO2_7.

9.5 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.

9.5.1 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 Section 4.4.41 “Pin Interrupt Select registers 0 to 7”). 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.



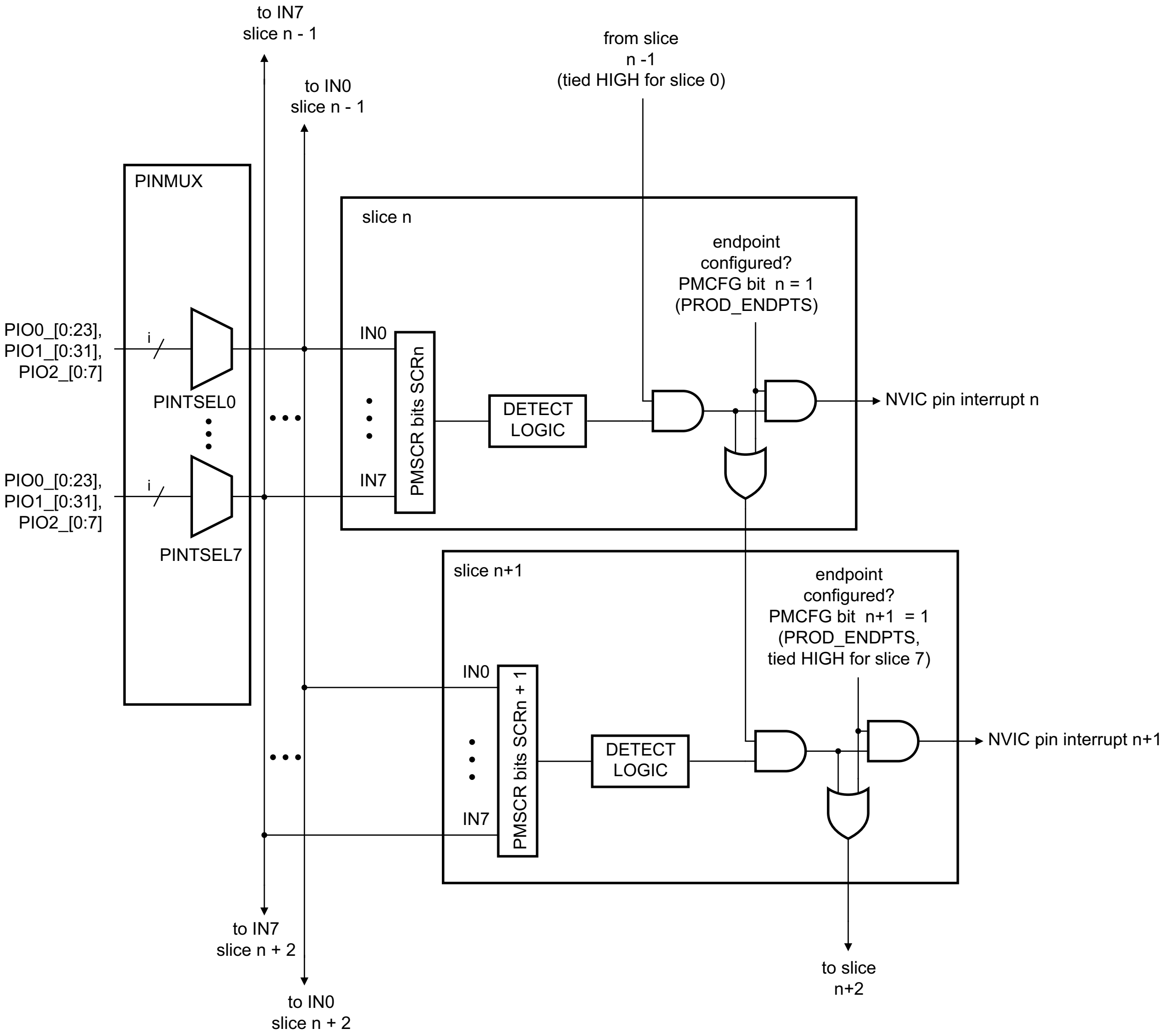
n = 6 for the DIP8 package, n= 14 for the TSSOP16 package, n = 18 for the TSSOP/SOP20 packages.

Fig 10. Pin interrupt connections

9.5.2 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 12 for the detect logic block.

Fig 11. 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 12 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 9.7.3 for details.

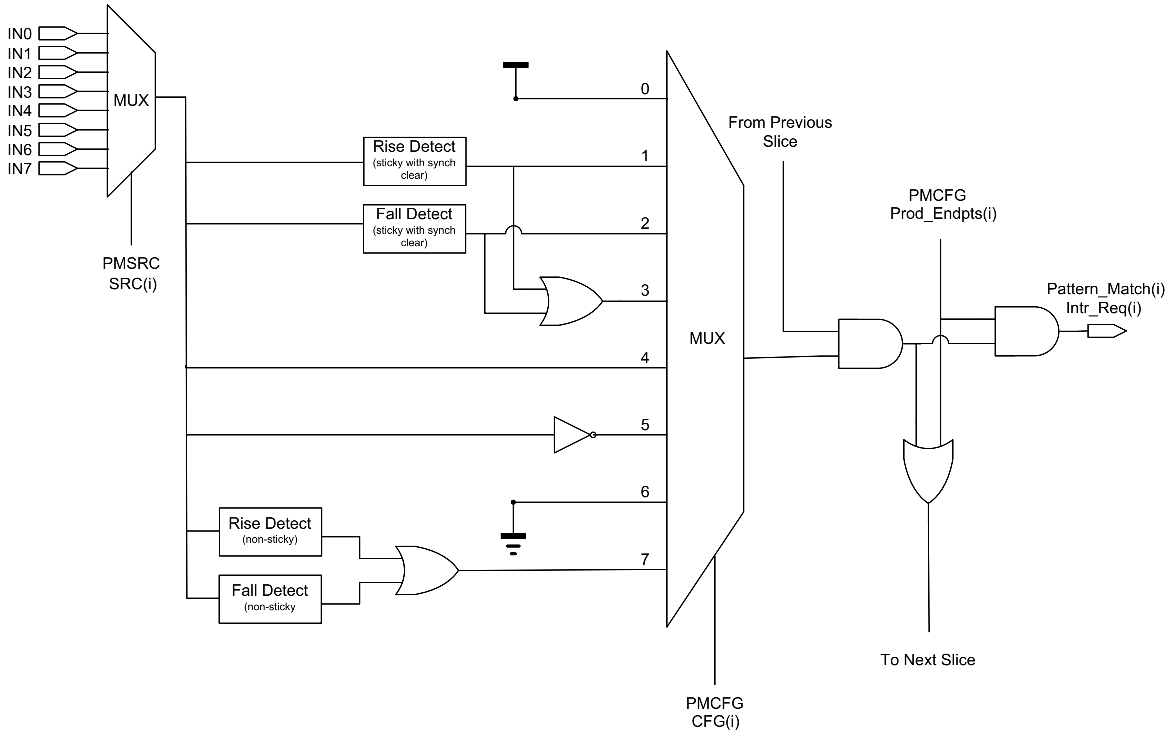


Fig 12. Pattern match bit slice with detect logic

9.5.2.1 Inputs and outputs of the pattern match engine

The connections between the pins and the pattern match engine are shown in Figure 11. 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 IOCON 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 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.

9.5.2.2 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 12).

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 118) and PMCFG (Table 119). 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. That is, if any of the three minterms are true, the output is asserted.

Related links:

Section 9.7.2

9.6 Register description

Table 106. Register overview: Pin interrupts/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 107
IENR	R/W	0x004	Pin interrupt level or rising edge interrupt enable register	0	Table 108
SIENR	WO	0x008	Pin interrupt level or rising edge interrupt set register	NA	Table 109
CIENR	WO	0x00C	Pin interrupt level (rising edge interrupt) clear register	NA	Table 110
IENF	R/W	0x010	Pin interrupt active level or falling edge interrupt enable register	0	Table 111
SIENF	WO	0x014	Pin interrupt active level or falling edge interrupt set register	NA	Table 112
CIENF	WO	0x018	Pin interrupt active level or falling edge interrupt clear register	NA	Table 113
RISE	R/W	0x01C	Pin interrupt rising edge register	0	Table 114
FALL	R/W	0x020	Pin interrupt falling edge register	0	Table 115
IST	R/W	0x024	Pin interrupt status register	0	Table 116
PMCTRL	R/W	0x028	Pattern match interrupt control register	0	Table 117
PMSRC	R/W	0x02C	Pattern match interrupt bit-slice source register	0	Table 118
PMCFG	R/W	0x030	Pattern match interrupt bit slice configuration register	0	Table 119

9.6.1 Pin interrupt mode register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 4.4.41 "Pin Interrupt Select registers 0 to 7"), one bit in the ISEL register determines whether the interrupt is edge or level sensitive.

Table 107. 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.	-	-

9.6.2 Pin interrupt level or rising edge interrupt enable register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 4.4.41 "Pin Interrupt Select registers 0 to 7"), 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 108. 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.	-	-

9.6.3 Pin interrupt level or rising edge interrupt set register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 4.4.41 “Pin Interrupt Select registers 0 to 7”), 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 109. 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.	-	-

9.6.4 Pin interrupt level or rising edge interrupt clear register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 4.4.41 “Pin Interrupt Select registers 0 to 7”), 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 110. 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.	-	-

9.6.5 Pin interrupt active level or falling edge interrupt enable register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 4.4.41 "Pin Interrupt Select registers 0 to 7"), 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 111. 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.	-	-

9.6.6 Pin interrupt active level or falling edge interrupt set register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 4.4.41 "Pin Interrupt Select registers 0 to 7"), 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 112. 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.	-	-

9.6.7 Pin interrupt active level or falling edge interrupt clear register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 4.4.41 “Pin Interrupt Select registers 0 to 7”), 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 113. 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.	-	-

9.6.8 Pin interrupt rising edge register

This register contains ones for pin interrupts selected in the PINTSELn registers (see Section 4.4.41 “Pin Interrupt Select registers 0 to 7”) 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 114. 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.	-	-

9.6.9 Pin interrupt falling edge register

This register contains ones for pin interrupts selected in the PINTSELn registers (see Section 4.4.41 “Pin Interrupt Select registers 0 to 7”) 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 115. 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.	-	-

9.6.10 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 116. 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.	-	-

9.6.11 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 ARM 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 117. 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 ARM CPU is disabled.	
		1	Enabled. RXEV output to the ARM 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

9.6.12 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 4.4.41 "Pin Interrupt Select registers 0 to 7". 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 118. 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
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.	

Table 118. Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description

Bit	Symbol	Value	Description	Reset value
13:11	SRC1		Selects the input source for bit slice 1	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 1.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 1.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 1.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 1.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 1.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 1.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 1.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 1.	
16:14	SRC2		Selects the input source for bit slice 2	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 2.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 2.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 2.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 2.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 2.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 2.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 2.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 2.	

Table 118. Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description

Bit	Symbol	Value	Description	Reset value
19:17	SRC3		Selects the input source for bit slice 3	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 3.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 3.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 3.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 3.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 3.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 3.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 3.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 3.	
22:20	SRC4		Selects the input source for bit slice 4	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 4.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 4.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 4.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 4.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 4.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 4.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 4.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 4.	

Table 118. Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description

Bit	Symbol	Value	Description	Reset value
25:23	SRC5		Selects the input source for bit slice 5	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 5.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 5.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 5.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 5.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 5.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 5.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 5.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 5.	
28:26	SRC6		Selects the input source for bit slice 6	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 6.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 6.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 6.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 6.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 6.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 6.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 6.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 6.	

Table 118. Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description

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

9.6.13 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 119. 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.	
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

Table 119. Pattern match bit slice configuration register (PMCFG, address 0xA000 4030) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
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 119. 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 119. 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 119. 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.	

9.7 Functional description

9.7.1 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 120.

Table 120. 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.

9.7.2 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 118):
  - 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 119):
  - 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 117):
  - 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.

9.7.3 Pattern match engine edge detect examples

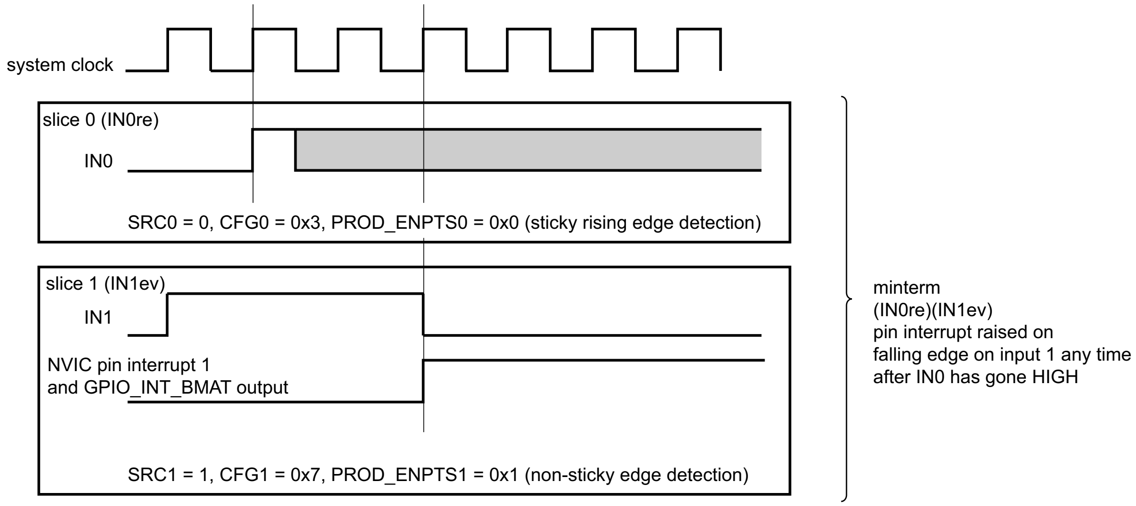


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 13. Pattern match engine examples: sticky edge detect

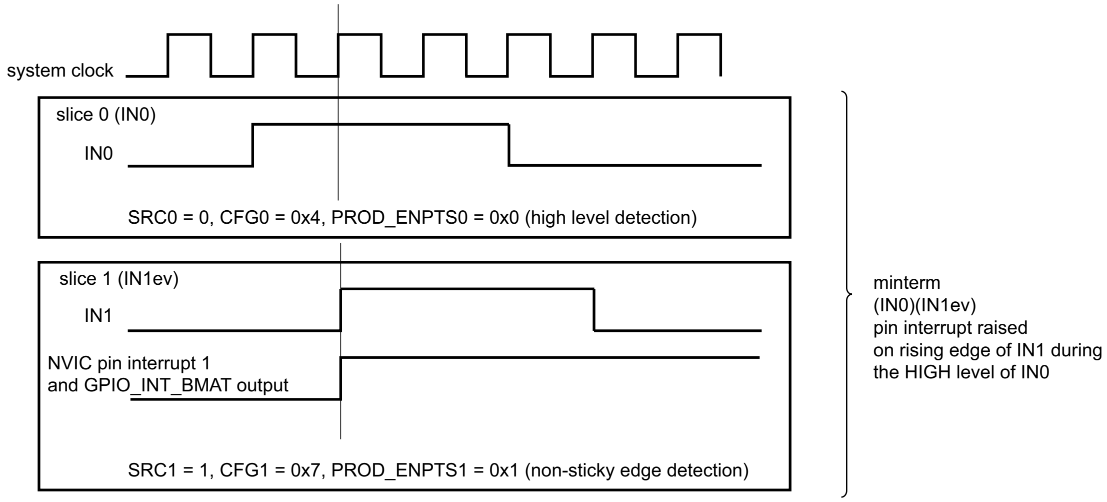


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 14. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true

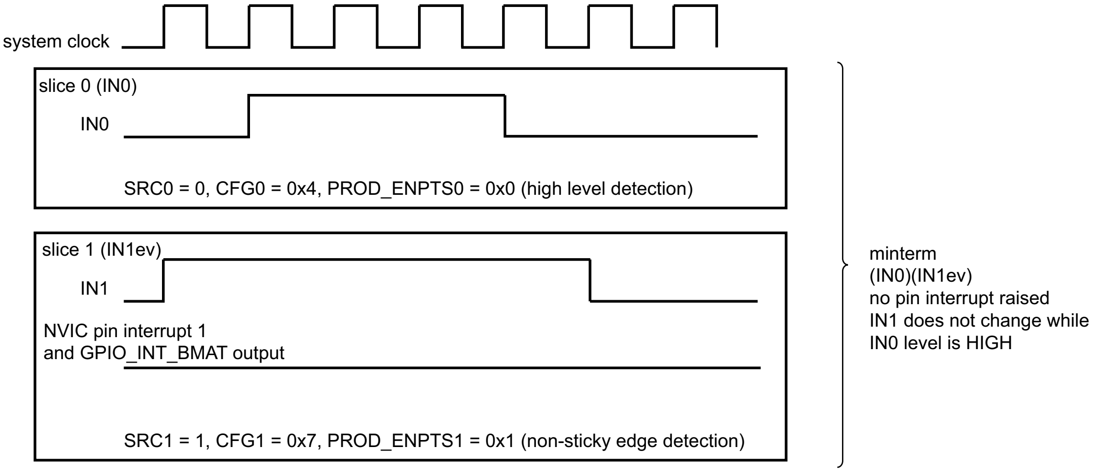


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

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

UM10732

Chapter 10: LPC11U6x/E6x DMA controller

Rev. 1.8 — 19 December 2016 **User manual**

10.1 How to read this chapter

The DMA controller is available on all parts.

10.2 Features

- 16 channels with 14 channels connected to peripheral request inputs.
- DMA operations can be triggered by on-chip events or two pin interrupts. Each DMA channel can select one trigger input from 12 sources.
- Priority is user selectable for each channel.
- Continuous priority arbitration.
- Address cache with two entries.
- Efficient use of data bus.
- Supports single transfers up to 1,024 words.
- Address increment options allow packing and/or unpacking data.

10.3 Basic configuration

Configure the DMA as follows:

- Use the SYSAHBCLKCTRL register (Table 40) to enable the clock to the DMA registers interface.
- Clear the DMA peripheral reset using the PRESETCTRL register (Table 23).
- The DMA interrupt is connected to slot #28 in the NVIC.
- Each DMA channel has one DMA request line associated and can also select one of 12 input triggers through the pinmux registers DMA_ITRIG_INMUX[0:15].

10.3.1 Input requests and triggers

Each DMA channel can use one input trigger that is independent of the request input for this channel. The trigger input is selected in the DMA_ITRIG_INMUX registers. There are 12 possible trigger sources, and each channel can select individually one of the 12 sources.

For each trigger DMA_ITRIG_INMUXn, the following sources are supported (see Table 148):

- 0 = ADC0_SEQA_IRQ
- 1 = ADC0_SEQB_IRQ
- 2 = CT16B0_MAT0
- 3 = CT16B1_MAT0
- 4 = CT32B0_MAT0
- 5 = CT16B1_MAT0

- 6 = PINT0 (pin interrupt 0)
- 7 = PINT1 (pin interrupt 1)
- 8 = SCT0_DMA0
- 9 = SCT0_DMA1
- 10 = SCT1_DMA0
- 11 = SCT1_DMA1

Table 121. DMA requests and triggers

DMA channel #	Request input	Trigger input mux	Reference
0	SSP0_RX_DMA	DMA_ITRIG_INMUX0	Table 148
1	SSP0_TX_DMA	DMA_ITRIG_INMUX1	Table 148
2	SSP1_RX_DMA	DMA_ITRIG_INMUX2	Table 148
3	SSP1_TX_DMA	DMA_ITRIG_INMUX3	Table 148
4	USART0_RX_DMA	DMA_ITRIG_INMUX4	Table 148
5	USART0_TX_DMA	DMA_ITRIG_INMUX5	Table 148
6	USART1_RX_DMA	DMA_ITRIG_INMUX6	Table 148
7	USART1_TX_DMA	DMA_ITRIG_INMUX7	Table 148
8	USART2_RX_DMA	DMA_ITRIG_INMUX8	Table 148
9	USART2_TX_DMA	DMA_ITRIG_INMUX9	Table 148
10	USART3_RX_DMA	DMA_ITRIG_INMUX10	Table 148
11	USART3_TX_DMA	DMA_ITRIG_INMUX11	Table 148
12	USART4_RX_DMA	DMA_ITRIG_INMUX12	Table 148
13	USART4_TX_DMA	DMA_ITRIG_INMUX13	Table 148
14	-	DMA_ITRIG_INMUX14	Table 148
15	-	DMA_ITRIG_INMUX15	Table 148

10.4 General description

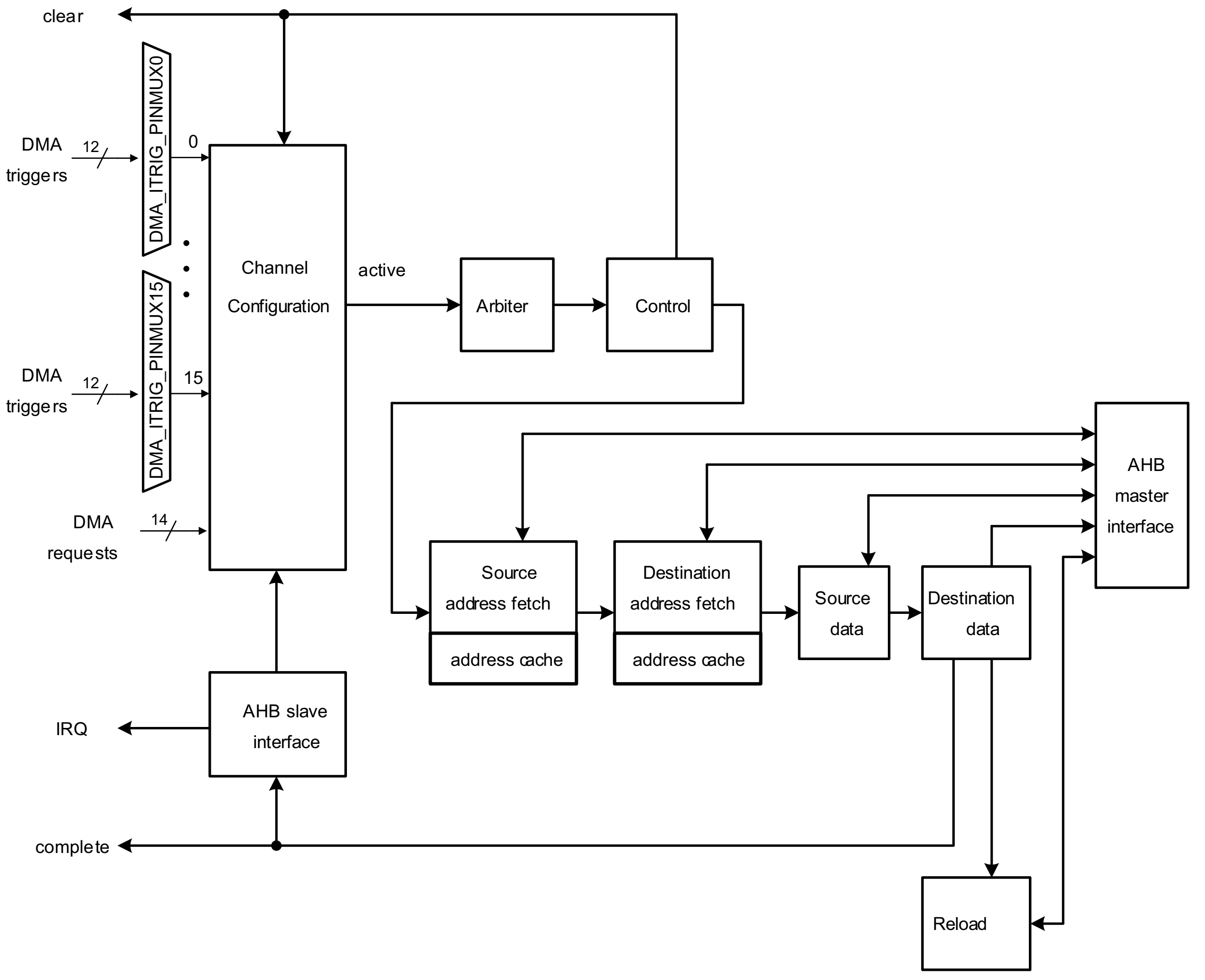


Fig 16. DMA block diagram

10.4.1 DMA requests and triggers

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

A trigger can be a signal from an unrelated peripheral, such as a timer or the ADC, that initiates a DMA operation. Triggers can be used to do things such as send a character or a string to a UART or other serial output at a fixed time interval or when an event occurs, possibly a timer match or an ADC sequence interrupt, or a GPIO pin changing state monitored by the PINT block.

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

10.4.2 DMA Modes

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

Once the DMA controller is set up for operation, using any specific DMA channel requires initializing the registers associated with that channel and supplying at least the channel descriptor, which is located somewhere in memory, typically in on-chip SRAM (see Section 10.5.3). The channel descriptor is shown in Table 122.

Table 122: Channel descriptor

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

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

After the channel has had a sufficient number of DMA requests and/or triggers, depending on its configuration, the initial descriptor will be exhausted. At that point, if the transfer configuration directs it, the channel descriptor will be reloaded with data from memory pointed to by the “Link to next descriptor” entry of the initial channel descriptor. Descriptors loaded in this manner look slightly different the channel descriptor, as shown in Table 123. The difference is that a new transfer configuration is specified in the reload descriptor instead of being written to the XFERCFG register for that channel.

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

Table 123: Reload descriptors

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

10.4.3 Single buffer

This generally applies to memory to memory moves, and peripheral DMA that occurs only occasionally and is set up for each transfer. For this kind of operation, only the initial channel descriptor shown in Table 124 is needed.

Table 124: Channel descriptor for a single transfer

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

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

10.4.4 Ping-Pong

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

A ping-pong transfer uses two buffers alternately. At any one time, one buffer is being loaded or unloaded by DMA operations. The other buffer has the opposite operation being handled by software, readying the buffer for use when the buffer currently being used by the DMA controller is full or empty. Table 125 shows an example of descriptors for ping-pong from a peripheral to two buffers in memory.

Table 125: Example descriptors for ping-pong operation: peripheral to buffer

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

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

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

10.4.5 Linked transfers (linked list)

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

An example of a linked transfer could start out like the example for a ping-pong transfer (Table 125). The difference would be that descriptor B would not link back to descriptor A, but would continue on to another different descriptor. This could continue as long as desired, and can be ended anywhere, or linked back to any point to repeat a sequence of descriptors. Of course, any descriptor not currently in use can be altered by software as well.

10.4.6 Address alignment for data transfers

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

10.5 Register description

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

In addition, the DMA trigger input on each channel is multiplexed. The input mux registers are located in the DMA TRIGMUX block.

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

Table 126. Register overview: DMA controller (base address 0x5000 4000)

Name	Access	Address offset	Description	Reset Value	Reference
Global control and status registers					
CTRL	R/W	0x000	DMA control.	0	Table 128
INTSTAT	RO	0x004	Interrupt status.	0	Table 129
SRAMBASE	R/W	0x008	SRAM address of the channel configuration table.	0	Table 130
Shared registers					
ENABLESET0	RO/W1	0x020	Channel Enable read and Set for all DMA channels.	0	Table 132
ENABLECLR0	W1	0x028	Channel Enable Clear for all DMA channels.	NA	Table 133
ACTIVE0	RO	0x030	Channel Active status for all DMA channels.	0	Table 134
BUSY0	RO	0x038	Channel Busy status for all DMA channels.	0	Table 135
ERRINT0	RO/W1	0x040	Error Interrupt status for all DMA channels.	0	Table 136
INTENSET0	RO/W1	0x048	Interrupt Enable read and Set for all DMA channels.	0	Table 137
INTENCLR0	W1	0x050	Interrupt Enable Clear for all DMA channels.	NA	Table 138
INTA0	RO/W1	0x058	Interrupt A status for all DMA channels.	0	Table 139
INTB0	RO/W1	0x060	Interrupt B status for all DMA channels.	0	Table 140
SETVALID0	W1	0x068	Set ValidPending control bits for all DMA channels.	NA	Table 141
SETTRIG0	W1	0x070	Set Trigger control bits for all DMA channels.	NA	Table 142
ABORT0	W1	0x078	Channel Abort control for all DMA channels.	NA	Table 143
Channel0 registers					
CFG0	R/W	0x400	Configuration register for DMA channel 0.		Table 144
CTLSTAT0	RO	0x404	Control and status register for DMA channel 0.		Table 146
XFERCFG0	R/W	0x408	Transfer configuration register for DMA channel 0.		Table 147
Channel1 registers					
CFG1	R/W	0x410	Configuration register for DMA channel 1.		Table 144
CTLSTAT1	RO	0x414	Control and status register for DMA channel 1.		Table 146
XFERCFG1	R/W	0x418	Transfer configuration register for DMA channel 1.		Table 147
Channel2 registers					
CFG2	R/W	0x420	Configuration register for DMA channel 2.		Table 144
CTLSTAT2	RO	0x424	Control and status register for DMA channel 2.		Table 146
XFERCFG2	R/W	0x428	Transfer configuration register for DMA channel 2.		Table 147
Channel3 registers					
CFG3	R/W	0x430	Configuration register for DMA channel 3.		Table 144
CTLSTAT3	RO	0x434	Control and status register for DMA channel 3.		Table 146
XFERCFG3	R/W	0x438	Transfer configuration register for DMA channel 3.		Table 147
Channel4 registers					
CFG4	R/W	0x440	Configuration register for DMA channel 4.		Table 144
CTLSTAT4	RO	0x444	Control and status register for DMA channel 4.		Table 146
XFERCFG4	R/W	0x448	Transfer configuration register for DMA channel 4.		Table 147
Channel5 registers					
CFG5	R/W	0x450	Configuration register for DMA channel 5.		Table 144
CTLSTAT5	RO	0x454	Control and status register for DMA channel 5.		Table 146

Table 126. Register overview: DMA controller (base address 0x5000 4000)

Name	Access	Address offset	Description	Reset Value	Reference
XFERCFG5	R/W	0x458	Transfer configuration register for DMA channel 5.		Table 147
Channel6 registers					
CFG6	R/W	0x460	Configuration register for DMA channel 6.		Table 144
CTLSTAT6	RO	0x464	Control and status register for DMA channel 6.		Table 146
XFERCFG6	R/W	0x468	Transfer configuration register for DMA channel 6.		Table 147
Channel7 registers					
CFG7	R/W	0x470	Configuration register for DMA channel 7.		Table 144
CTLSTAT7	RO	0x474	Control and status register for DMA channel 7.		Table 146
XFERCFG7	R/W	0x478	Transfer configuration register for DMA channel 7.		Table 147
Channel8 registers					
CFG8	R/W	0x480	Configuration register for DMA channel 8.		Table 144
CTLSTAT8	RO	0x484	Control and status register for DMA channel 8.		Table 146
XFERCFG8	R/W	0x488	Transfer configuration register for DMA channel 8.		Table 147
Channel9 registers					
CFG9	R/W	0x490	Configuration register for DMA channel 9.		Table 144
CTLSTAT9	RO	0x494	Control and status register for DMA channel 9.		Table 146
XFERCFG9	R/W	0x498	Transfer configuration register for DMA channel 9.		Table 147
Channel10 registers					
CFG10	R/W	0x4A0	Configuration register for DMA channel 10.		Table 144
CTLSTAT10	RO	0x4A4	Control and status register for DMA channel 10.		Table 146
XFERCFG10	R/W	0x4A8	Transfer configuration register for DMA channel 10.		Table 147
Channel11 registers					
CFG11	R/W	0x4B0	Configuration register for DMA channel 11.		Table 144
CTLSTAT11	RO	0x4B4	Control and status register for DMA channel 11.		Table 146
XFERCFG11	R/W	0x4B8	Transfer configuration register for DMA channel 11.		Table 147
Channel12 registers					
CFG12	R/W	0x4C0	Configuration register for DMA channel 12.		Table 144
CTLSTAT12	RO	0x4C4	Control and status register for DMA channel 12.		Table 146
XFERCFG12	R/W	0x4C8	Transfer configuration register for DMA channel 12.		Table 147
Channel13 registers					
CFG13	R/W	0x4D0	Configuration register for DMA channel 13.		Table 144
CTLSTAT13	RO	0x4D4	Control and status register for DMA channel 13.		Table 146
XFERCFG13	R/W	0x4D8	Transfer configuration register for DMA channel 13.		Table 147
Channel14 registers					
CFG14	R/W	0x4E0	Configuration register for DMA channel 14.		Table 144
CTLSTAT14	RO	0x4E4	Control and status register for DMA channel 14.		Table 146
XFERCFG14	R/W	0x4E8	Transfer configuration register for DMA channel 14.		Table 147
Channel15 registers					

Table 126. Register overview: DMA controller (base address 0x5000 4000)

Name	Access	Address offset	Description	Reset Value	Reference
CFG15	R/W	0x4F0	Configuration register for DMA channel 15.		Table 144
CTLSTAT15	RO	0x4F4	Control and status register for DMA channel 15.		Table 146
XFERCFG15	R/W	0x4F8	Transfer configuration register for DMA channel 15.		Table 147

Table 127. Register overview: Pin multiplexing DMA TRIGMUX (base address 0x4002 8000)

Name	Access	Offset	Description	Reset value	Reset value after boot	Reference
DMA_ITRIG_INMUX0	R/W	0x000	Trigger input select register for DMA channel 0.	0x1F		Table 148
DMA_ITRIG_INMUX1	R/W	0x004	Trigger input select register for DMA channel 1.	0x1F		Table 148
DMA_ITRIG_INMUX2	R/W	0x008	Trigger input select register for DMA channel 2.	0x1F		Table 148
DMA_ITRIG_INMUX3	R/W	0x00C	Trigger input select register for DMA channel 3.	0x1F		Table 148
DMA_ITRIG_INMUX4	R/W	0x010	Trigger input select register for DMA channel 4.	0x1F		Table 148
DMA_ITRIG_INMUX5	R/W	0x014	Trigger input select register for DMA channel 5.	0x1F		Table 148
DMA_ITRIG_INMUX6	R/W	0x018	Trigger input select register for DMA channel 6.	0x1F		Table 148
DMA_ITRIG_INMUX7	R/W	0x01C	Trigger input select register for DMA channel 7.	0x1F		Table 148
DMA_ITRIG_INMUX8	R/W	0x020	Trigger input select register for DMA channel 8.	0x1F		Table 148
DMA_ITRIG_INMUX9	R/W	0x024	Trigger input select register for DMA channel 9.	0x1F		Table 148
DMA_ITRIG_INMUX10	R/W	0x028	Trigger input select register for DMA channel 10.	0x1F		Table 148
DMA_ITRIG_INMUX11	R/W	0x02C	Trigger input select register for DMA channel 11.	0x1F		Table 148
DMA_ITRIG_INMUX12	R/W	0x030	Trigger input select register for DMA channel 12.	0x1F		Table 148
DMA_ITRIG_INMUX13	R/W	0x034	Trigger input select register for DMA channel 13.	0x1F		Table 148
DMA_ITRIG_INMUX14	R/W	0x038	Trigger input select register for DMA channel 14.	0x1F		Table 148
DMA_ITRIG_INMUX15	R/W	0x03C	Trigger input select register for DMA channel 15.	0x1F		Table 148

10.5.1 Control register

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

Table 128. Control register (CTRL, address 0x5000 4000) bit description

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

10.5.2 Interrupt Status register

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

Table 129. Interrupt Status register (INTSTAT, address 0x5000 4004) bit description

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

10.5.3 SRAM Base address register

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

Table 130. SRAM Base address register (SRAMBASE, address 0x5000 4008) bit description

Bit	Symbol	Description	Reset value
7:0	-	Reserved. Read value is undefined, only zero should be written.	NA
31:8	OFFSET	Address bits 31:8 of the beginning of the DMA descriptor table. For 16 channels, the table must begin on a 256 byte boundary.	0

Each DMA channel has an entry for the channel descriptor in the SRAM table. The values for each channel start at the address offsets found in Table 131. Only the descriptors for channels defined at extraction are used. The contents of each channel descriptor are described in Table 122.

Table 131. Channel descriptor map

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

10.5.4 Enable read and Set registers

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

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

Table 132. Enable read and Set register 0 (ENABLESET0, address 0x5000 4020) bit description

Bit	Symbol	Description	Reset value
15:0	ENA	Enable for DMA channels 15:0. Bit n enables or disables DMA channel n. 0 = disabled. 1 = enabled.	0
31:16	-	Reserved.	

10.5.5 Enable Clear register

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

Table 133. Enable Clear register 0 (ENABLECLR0, address 0x5000 4028) bit description

Bit	Symbol	Description	Reset value
15:0	CLR	Writing ones to this register clears the corresponding bits in ENABLESET0. Bit n clears the channel enable bit n.	NA
31:16			

10.5.6 Active status register

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

A DMA channel is considered active when a DMA operation has been started but not yet fully completed. The Active status will persist from a DMA operation being started, until the pipeline is empty after end of the last descriptor (when there is no reload). An active channel may be aborted by software by setting the appropriate bit in one of the Abort register (see Section 10.5.15).

Table 134. Active status register 0 (ACTIVE0, address 0x5000 4030) bit description

Bit	Symbol	Description	Reset value
15:0	ACT	Active flag for DMA channel n. Bit n corresponds to DMA channel n. 0 = not active. 1 = active.	0
31:16	-	Reserved.	-

10.5.7 Busy status register

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

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

Table 135. Busy status register 0 (BUSY0, address 0x5000 4038) bit description

Bit	Symbol	Description	Reset value
15:0	BSY	Busy flag for DMA channel n. Bit n corresponds to DMA channel n. 0 = not busy. 1 = busy.	0
31:16	-	Reserved.	-

10.5.8 Error Interrupt register

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

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

Table 136. Error Interrupt register 0 (ERRINT0, address 0x5000 4040) bit description

Bit	Symbol	Description	Reset value
15:0	ERR	Error Interrupt flag for DMA channel n. Bit n corresponds to DMA channel n. 0 = error interrupt is not active. 1 = error interrupt is active.	0
31:16	-	Reserved.	-

10.5.9 Interrupt Enable read and Set register

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

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

Table 137. Interrupt Enable read and Set register 0 (INTENSET0, address 0x5000 4048) bit description

Bit	Symbol	Description	Reset value
15: 0	INTEN	Interrupt Enable read and set for DMA channel n. Bit n corresponds to DMA channel n. 0 = interrupt for DMA channel is disabled. 1 = interrupt for DMA channel is enabled.	0
31:16	-	Reserved.	-

10.5.10 Interrupt Enable Clear register

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

Table 138. Interrupt Enable Clear register 0 (INTENCLR0, address 0x5000 4050) bit description

Bit	Symbol	Description	Reset value
15:0	CLR	Writing ones to this register clears corresponding bits in the INTENSET0. Bit n corresponds to DMA channel n.	NA
31:16	-	Reserved.	-

10.5.11 Interrupt A register

The IntA0 and IntA1 register contains the interrupt A status for each DMA channel. The status will be set when the SETINTA bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in these registers clears the related INTA flag. Writing 0 has no effect. Any interrupt pending status in this register will be reflected on the DMA interrupt output if it is enabled in the related INTENSET register.

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

Table 139. Interrupt A register 0 (INTA0, address 0x5000 4058) bit description

Bit	Symbol	Description	Reset value
15:0	IA	Interrupt A status for DMA channel n. Bit n corresponds to DMA channel n. 0 = the DMA channel interrupt A is not active. 1 = the DMA channel interrupt A is active.	0
31:16	-	Reserved.	-

10.5.12 Interrupt B register

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

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

Table 140. Interrupt B register 0 (INTB0, address 0x5000 4060) bit description

Bit	Symbol	Description	Reset value
15:0	IB	Interrupt B status for DMA channel n. Bit n corresponds to DMA channel n. 0 = the DMA channel interrupt B is not active. 1 = the DMA channel interrupt B is active.	0
31:16	-	Reserved.	-

10.5.13 Set Valid register

The SETVALID0 register allows setting the Valid bit in the CTRLSTAT register for one or more DMA channel. See Section 10.5.17 for a description of the VALID bit.

The CFGVALID and SV (set valid) bits allow more direct DMA block timing control by software. Each Channel Descriptor, in a sequence of descriptors, can be validated by either the setting of the CFGVALID bit or by setting the channel's SETVALID flag. Normally, the CFGVALID bit is set. This tells the DMA that the Channel Descriptor is active and can be executed. The DMA will continue sequencing through descriptor blocks whose CFGVALID bit are set without further software intervention. Leaving a CFGVALID bit set to 0 allows the DMA sequence to pause at the Descriptor until software triggers the continuation. If, during DMA transmission, a Channel Descriptor is found with CFGVALID set to 0, the DMA checks for a previously buffered SETVALID0 setting for the channel. If found, the DMA will set the descriptor valid, clear the SV setting, and resume processing the descriptor. Otherwise, the DMA pauses until the channels SETVALID0 bit is set.

Table 141. Set Valid 0 register (SETVALID0, address 0x5000 4068) bit description

Bit	Symbol	Description	Reset value
15:0	SV	SETVALID control for DMA channel n. Bit n corresponds to DMA channel n. 0 = no effect. 1 = sets the VALIDPENDING control bit for DMA channel n.	NA
31:16	-	Reserved.	-

10.5.14 Set Trigger register

The SETTRIG0 register allows setting the TRIG bit in the CTRLSTAT register for one or more DMA channel. See Section 10.5.17 for a description of the TRIG bit, and Section 10.4.1 for a general description of triggering.

Table 142. Set Trigger 0 register (SETTRIG0, address 0x5000 4070) bit description

Bit	Symbol	Description	Reset value
15:0	TRIG	Set Trigger control bit for DMA channel 0. Bit n corresponds to DMA channel n. 0 = no effect. 1 = sets the TRIG bit for DMA channel n.	NA
31:16	-	Reserved.	-

10.5.15 Abort registers

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

Table 143. Abort 0 register (ABORT0, address 0x5000 4078) bit description

Bit	Symbol	Description	Reset value
15:0	AORTCTRL	Abort control for DMA channel 0. Bit n corresponds to DMA channel n. 0 = no effect. 1 = aborts DMA operations on channel n.	NA
31:16	-	Reserved.	-

10.5.16 Channel configuration registers

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

See Table 145 for a summary of trigger options.

Table 144. Configuration registers for channel 0 to 15 (CFG[0:15], addresses 0x5000 4400 (CFG0) to address 0x5000 44F0 (CFG15)) bit description

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

Table 144. Configuration registers for channel 0 to 15 (CFG[0:15], addresses 0x5000 4400 (CFG0) to address 0x5000 44F0 (CFG15)) bit description

Bit	Symbol	Value	Description	Reset value
11:8	BURSTPOWER		Burst Power is used in two ways. It always selects the address wrap size when SRCBURSTWRAP and/or DSTBURSTWRAP modes are selected. When the TRIGBURST field elsewhere in this register = 1, Burst Power selects how many transfers are performed for each DMA trigger. This can be used, for example, with peripherals that contain a FIFO that can initiate a DMA operation when the FIFO reaches a certain level. 0000: Burst size = 1 (2^0). 0001: Burst size = 2 (2^1). 0010: Burst size = 4 (2^2). ... 1010: Burst size = 1024 (2^{10}). This corresponds to the maximum supported transfer count. others: not supported. The total transfer length as defined in the XFERCOUNT bits in the XFERCFG register must be an even multiple of the burst size.	0
13:12	-		Reserved. Read value is undefined, only zero should be written.	NA
14	SRCBURSTWRAP		Source Burst Wrap. When enabled, the source data address for the DMA is “wrapped”, meaning that the source address range for each burst will be the same. As an example, this could be used to read several sequential registers from a peripheral for each DMA burst, reading the same registers again for each burst.	0
		0	Disabled. Source burst wrapping is not enabled for this DMA channel.	
		1	Enabled. Source burst wrapping is enabled for this DMA channel.	
15	DSTBURSTWRAP		Destination Burst Wrap. When enabled, the destination data address for the DMA is “wrapped”, meaning that the destination address range for each burst will be the same. As an example, this could be used to write several sequential registers to a peripheral for each DMA burst, writing the same registers again for each burst.	0
		0	Disabled. Destination burst wrapping is not enabled for this DMA channel.	
		1	Enabled. Destination burst wrapping is enabled for this DMA channel.	
17:16	CHPRIORITY		Priority of this channel when multiple DMA requests are pending. 0x0 = highest priority. 0x3 = lowest priority.	0
31:18	-		Reserved. Read value is undefined, only zero should be written.	NA

Table 145. Trigger setting summary

TRIGBURST	TRIGTYPE	TRIGPOL	Description
0	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SRCBURSTWRAP and/or DSTBURSTWRAP.
0	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SRCBURSTWRAP and/or DSTBURSTWRAP.
0	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SRCBURSTWRAP and/or DSTBURSTWRAP.

Table 145. Trigger setting summary

TRIGBURST	TRIGTYPE	TRIGPOL	Description
0	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SRCBURSTWRAP and/or DSTBURSTWRAP.
1	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SRCBURSTWRAP and/or DSTBURSTWRAP, and also determines how much data is transferred for each trigger.
1	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SRCBURSTWRAP and/or DSTBURSTWRAP, and also determines how much data is transferred for each trigger.
1	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SRCBURSTWRAP and/or DSTBURSTWRAP, and also determines how much data is transferred for each trigger.
1	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SRCBURSTWRAP and/or DSTBURSTWRAP, and also determines how much data is transferred for each trigger.

10.5.17 Channel control and status registers

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

Table 146. Control and Status registers for channel 0 to 15 (CTLSTAT[0:15], 0x5000 4404 (CTLSTAT0) to address 0x5000 44F4 (CTLSTAT15)) bit description

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

10.5.18 Channel transfer configuration registers

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

See “Trigger operation” for details on trigger operation.

Table 147. Transfer Configuration registers for channel 0 to 15 (XFERCFG[0:15], addresses 0x5000 4408 (XFERCFG0) to 0x5000 44F8 (XFERCFG15)) bit description

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

Table 147. Transfer Configuration registers for channel 0 to 15 (XFERCFG[0:15], addresses 0x5000 4408 (XFERCFG0) to 0x5000 44F8 (XFERCFG15)) bit description

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

10.5.19 DMA trigger input mux registers 0 to 15

With the DMA trigger input mux registers you can select one trigger input for each of the 16 DMA channels from 20 internal sources.

By default, none of the triggers are selected.

Table 148. DMA trigger input mux registers 0 to 15 (DMA_ITRIG_INMUX[0:15], address 0x4002 80E0 (DMA_ITRIG_INMUX0) to 0x4002 811C (DMA_ITRIG_INMUX15)) bit description

Bit	Symbol	Description	Reset value
3:0	INP_N	Trigger input number (decimal value) to DMA channel n. All other values are reserved. 0 = ADC0_SEQA_IRQ 1 = ADC0_SEQB_IRQ 2 = CT16B0_MAT0 3 = CT16B1_MAT0 4 = CT32B0_MAT0 5 = CT32B1_MAT0 6 = PINT0 (pin interrupt 0) 7 = PINT1 (pin interrupt 1) 8 = SCT0_DMA0 9 = SCT0_DMA1 10 = SCT1_DMA0 11 = SCT1_DMA1	0xF
31:4	-	Reserved.	-

10.6 Functional description

10.6.1 Trigger operation

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

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

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

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

UM10732

Chapter 11: LPC11U6x/E6x USART0

Rev. 1.8 — 19 December 2016 **User manual**

11.1 How to read this chapter

The USART0 is available on all parts.

Remark: The USART0 register map and register functions are different from the register map and register functions of the USART1 to USART4 peripherals.

11.2 Features

- 16-byte receive and transmit FIFOs.
- Register locations conform to ‘550 industry standard.
- Receiver FIFO trigger points at 1, 4, 8, and 14 bytes.
- Built-in baud rate generator.
- Software or hardware flow control.
- RS-485/EIA-485 9-bit mode support with output enable.
- $\overline{\text{RTS}}$/$\overline{\text{CTS}}$ flow control and other modem control signals.
- 1X-clock send or receive.
- ISO 7816-3 compliant smart card interface.
- IrDA support.
- DMA support.

11.3 Basic configuration

USART0 is configured using the following registers:

- In the SYSAHBCLKCTRL register, set bit 12 (Table 40) to enable the clock to the register interface.
- The USART0 peripheral clock PCLK is derived from the main clock divided by the USART0 peripheral clock divider (Table 42).
- Baud rate: In register LCR (Table 159), set bit DLAB =1. This enables access to registers DLL (Table 153) and DLM (Table 154) for setting the baud rate. Also, if needed, set the fractional baud rate in the fractional divider register (Table 167).
- UART FIFO: Use bit FIFO enable (bit 0) in register FCR (Table 158) to enable the FIFOs.
- Pins: Select UART pins and pin modes through the relevant IOCON registers (see Table 83).
- Interrupts: To enable USART0 interrupts set bit DLAB =0 in register LCR (Table 159). This enables access to IER (Table 155). Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
- DMA: USART0 transmit and receive functions can operate with the DMA controller (see Table 121).

Remark: USART0 cannot wake up the part from deep-sleep, power-down or deep power-down modes.

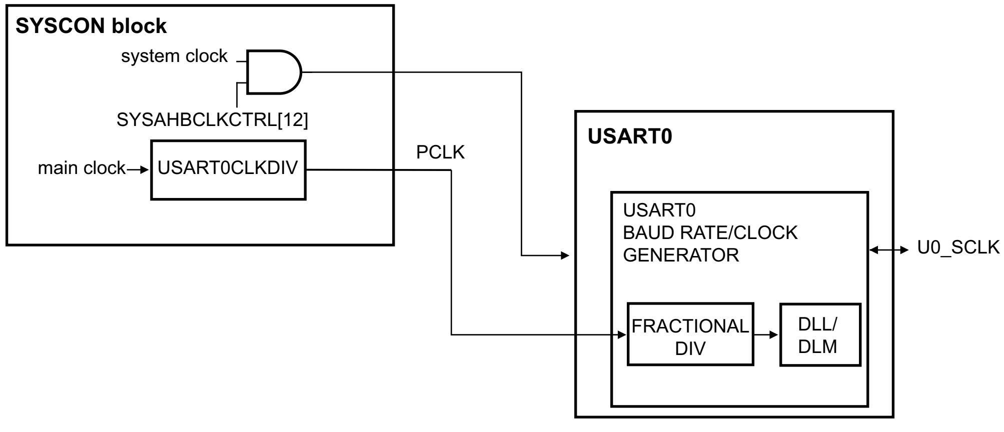


Fig 17. USART0 clocking

11.4 General description

The architecture of USART0 is shown below in the block diagram.

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

The USART0 receiver block, RX, monitors the serial input line, RXDn, for valid input. The USART0 RX Shift Register (RSR) accepts valid characters via RXDn. After a valid character is assembled in RSR, it is passed to the USART0 RX Buffer Register FIFO to await access by the CPU or host via the generic host interface.

The USART0 transmitter block, TX, accepts data written by the CPU or host and buffers the data in the USART0 TX Holding Register FIFO (THR). The USART0 TX Shift Register (TSR) reads the data stored in THR and assembles the data to transmit via the serial output pin, TXDn.

The USART0 Baud Rate Generator block, BRG, generates the timing enables used by the USART0 TX block. The BRG clock input source is the APB clock (PCLK). The main clock is divided down per the divisor specified in the DLL and DLM registers. This divided down clock is the 16x oversample clock.

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

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

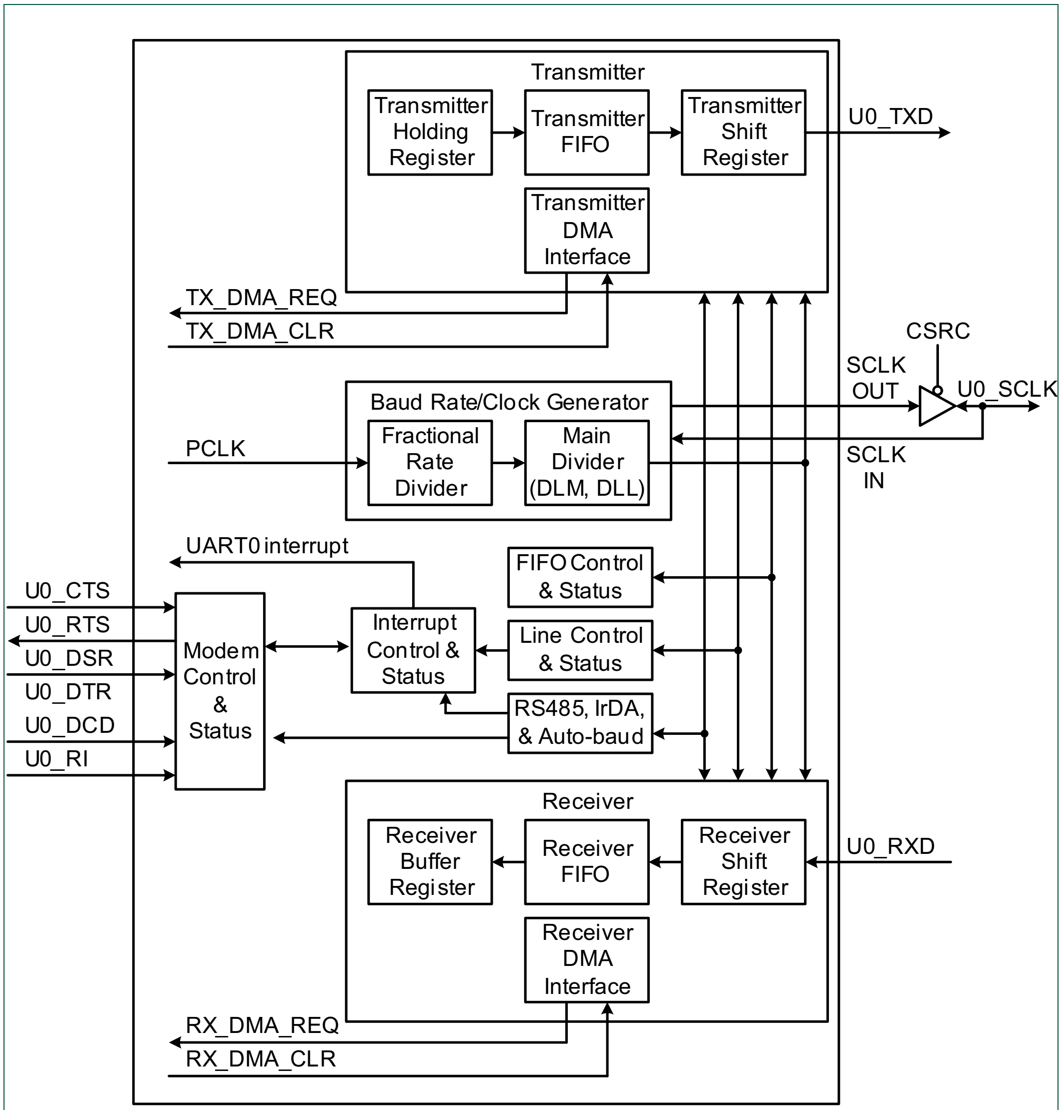


Fig 18. USART0 block diagram

11.5 Pin description

Table 149. USART0 pin description

Pin	Type	Description
U0_RXD	Input	Serial Input. Serial receive data.
U0_TXD	Output	Serial Output. Serial transmit data (input/output in smart card mode).
$\overline{\text{U0\_RTS}}$	Output	Request To Send. RS-485 direction control pin.
$\overline{\text{U0\_CTS}}$	Input	Clear To Send.
$\overline{\text{U0\_DTR}}$	Output	Data Terminal Ready.
$\overline{\text{U0\_DSR}}$	Input	Data Set Ready.

Table 149. USART0 pin description

Pin	Type	Description
$\overline{\text{U0\_DCD}}$	Input	Data Carrier Detect.
$\overline{\text{U0\_RI}}$	Input	Ring Indicator.
U0_SCLK	I/O	Serial Clock.

11.6 Register description

The USART0 contains registers organized as shown in Table 150. The Divisor Latch Access Bit (DLAB) is contained in the LCR register bit 7 and enables access to the Divisor Latches.

Offsets/addresses not shown in Table 150 are reserved.

Table 150. Register overview: USART0 (base address: 0x4000 8000)

Name	Access	Address offset	Description	Reset value[1]	Reference
RBR	RO	0x000	Receiver Buffer Register. Contains the next received character to be read. (DLAB=0)	NA	Table 151
THR	WO	0x000	Transmit Holding Register. The next character to be transmitted is written here. (DLAB=0)	NA	Table 152
DLL	R/W	0x000	Divisor Latch LSB. Least significant byte of the baud rate divisor value. The full divisor is used to generate a baud rate from the fractional rate divider. (DLAB=1)	0x01	Table 153
DLM	R/W	0x004	Divisor Latch MSB. Most significant byte of the baud rate divisor value. The full divisor is used to generate a baud rate from the fractional rate divider. (DLAB=1)	0	Table 154
IER	R/W	0x004	Interrupt Enable Register. Contains individual interrupt enable bits for the 7 potential USART0 interrupts. (DLAB=0)	0	Table 155
IIR	RO	0x008	Interrupt ID Register. Identifies which interrupt(s) are pending.	0x01	Table 156
FCR	WO	0x008	FIFO Control Register. Controls USART0 FIFO usage and modes.	0	Table 158
LCR	R/W	0x00C	Line Control Register. Contains controls for frame formatting and break generation.	0	Table 159
MCR	R/W	0x010	Modem Control Register.	0	Table 160
LSR	RO	0x014	Line Status Register. Contains flags for transmit and receive status, including line errors.	0x60	Table 161
MSR	RO	0x018	Modem Status Register.	0	Table 162
SCR	R/W	0x01C	Scratch Pad Register. Eight-bit temporary storage for software.	0	Table 163
ACR	R/W	0x020	Auto-baud Control Register. Contains controls for the auto-baud feature.	0	Table 164
ICR	R/W	0x024	IrDA Control Register. Enables and configures the IrDA (remote control) mode.	0	Table 165
FDR	R/W	0x028	Fractional Divider Register. Generates a clock input for the baud rate divider.	0x10	Table 167

Table 150. Register overview: USART0 (base address: 0x4000 8000)

Name	Access	Address offset	Description	Reset value[1]	Reference
OSR	R/W	0x02C	Oversampling Register. Controls the degree of oversampling during each bit time.	0xF0	Table 168
TER	R/W	0x030	Transmit Enable Register. Turns off USART0 transmitter for use with software flow control.	0x80	Table 169
HDEN	R/W	0x040	Half duplex enable register.	0	Table 170
SCICTRL	R/W	0x048	Smart Card Interface Control register. Enables and configures the Smart Card Interface feature.	0	Table 171
RS485CTRL	R/W	0x04C	RS-485/EIA-485 Control. Contains controls to configure various aspects of RS-485/EIA-485 modes.	0	Table 172
RS485ADRMATCH	R/W	0x050	RS-485/EIA-485 address match. Contains the address match value for RS-485/EIA-485 mode.	0	Table 173
RS485DLY	R/W	0x054	RS-485/EIA-485 direction control delay.	0	Table 174
SYNCCTRL	R/W	0x058	Synchronous mode control register.	0	Table 175

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

11.6.1 USART0 Receiver Buffer Register (when DLAB = 0, Read Only)

The RBR is the top byte of the USART0 RX FIFO. The top byte of the RX FIFO contains the oldest character received and can be read via the bus interface. The LSB (bit 0) contains the first-received data bit. If the character received is less than 8 bits, the unused MSBs are padded with zeros.

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

Since PE, FE and BI bits (see Table 161) correspond to the byte on the top of the RBR FIFO (i.e. the one that will be read in the next read from the RBR), the right approach for fetching the valid pair of received byte and its status bits is first to read the content of the LSR register, and then to read a byte from the RBR.

Table 151. USART0 Receiver Buffer Register when DLAB = 0, Read Only (RBR, address 0x4000 8000) bit description

Bit	Symbol	Description	Reset Value
7:0	RBR	The USART0 Receiver Buffer Register contains the oldest received byte in the USART0 RX FIFO.	undefined
31:8	-	Reserved	-

11.6.2 USART0 Transmitter Holding Register (when DLAB = 0, Write Only)

The THR is the top byte of the USART0 TX FIFO. The top byte is the newest character in the TX FIFO and can be written via the bus interface. The LSB represents the first bit to transmit.

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

Table 152. USART0 Transmitter Holding Register when DLAB = 0, Write Only (THR, address 0x4000 8000) bit description

Bit	Symbol	Description	Reset Value
7:0	THR	Writing to the USART0 Transmit Holding Register causes the data to be stored in the USART0 transmit FIFO. The byte will be sent when it is the oldest byte in the FIFO and the transmitter is available.	NA
31:8	-	Reserved	-

11.6.3 USART0 Divisor Latch LSB and MSB Registers (when DLAB = 1)

The USART0 Divisor Latch is part of the USART0 Baud Rate Generator and holds the value used (optionally with the Fractional Divider) to divide the UART_PCLK clock in order to produce the baud rate clock, which must be the multiple of the desired baud rate that is specified by the Oversampling Register (typically 16X). The DLL and DLM registers together form a 16-bit divisor. DLL contains the lower 8 bits of the divisor and DLM contains the higher 8 bits. A zero value is treated like 0x0001. The Divisor Latch Access Bit (DLAB) in the LCR must be one in order to access the USART0 Divisor Latches. Details on how to select the right value for DLL and DLM can be found in Section 11.6.14.

Table 153. USART0 Divisor Latch LSB Register when DLAB = 1 (DLL, address 0x4000 8000) bit description

Bit	Symbol	Description	Reset value
7:0	DLLSB	The USART0 Divisor Latch LSB Register, along with the DLM register, determines the baud rate of the USART0.	0x01
31:8	-	Reserved	-

Table 154. USART0 Divisor Latch MSB Register when DLAB = 1 (DLM, address 0x4000 8004) bit description

Bit	Symbol	Description	Reset value
7:0	DLMSB	The USART0 Divisor Latch MSB Register, along with the DLL register, determines the baud rate of the USART0.	0x00
31:8	-	Reserved	-

11.6.4 USART0 Interrupt Enable Register (when DLAB = 0)

The IER is used to enable the various USART0 interrupt sources.

Table 155. USART0 Interrupt Enable Register when DLAB = 0 (IER, address 0x4000 8004) bit description

Bit	Symbol	Value	Description	Reset value
0	RBRINTEN		RBR Interrupt Enable. Enables the Receive Data Available interrupt. It also controls the Character Receive Time-out interrupt.	0
		0	Disable. Disable the RDA interrupt.	
		1	Enable. Enable the RDA interrupt.	

Table 155. USART0 Interrupt Enable Register when DLAB = 0 (IER, address 0x4000 8004) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
1	THREINTEN		THRE Interrupt Enable. Enables the THRE interrupt. The status of this interrupt can be read from LSR[5].	0
		0	Disable. Disable the THRE interrupt.	
		1	Enable. Enable the THRE interrupt.	
2	RLSINTEN		Enables the Receive Line Status interrupt. The status of this interrupt can be read from LSR[4:1].	-
		0	Disable. Disable the RLS interrupt.	
		1	Enable. Enable the RLS interrupt.	
3	MSINTEN		Enables the Modem Status interrupt. The components of this interrupt can be read from the MSR.	
		0	Disable. Disable the MS interrupt.	
		1	Enable. Enable the MS interrupt.	
7:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	ABEOINTEN		Enables the end of auto-baud interrupt.	0
		0	Disable. Disable end of auto-baud Interrupt.	
		1	Enable. Enable end of auto-baud Interrupt.	
9	ABTOINTEN		Enables the auto-baud time-out interrupt.	0
		0	Disable. Disable auto-baud time-out Interrupt.	
		1	Enable. Enable auto-baud time-out Interrupt.	
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

11.6.5 USART0 Interrupt Identification Register (Read Only)

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

Table 156. USART0 Interrupt Identification Register Read only (IIR, address 0x4000 8008) bit description

Bit	Symbol	Value	Description	Reset value
0	INTSTATUS		Interrupt status. Note that IIR[0] is active low. The pending interrupt can be determined by evaluating IIR[3:1].	1
		0	Interrupt pending. At least one interrupt is pending.	
		1	Not pending. No interrupt is pending.	

Table 156. USART0 Interrupt Identification Register Read only (IIR, address 0x4000 8008) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
3:1	INTID		Interrupt identification. IER[3:1] identifies an interrupt corresponding to the USART0 Rx FIFO. All other values of IER[3:1] not listed below are reserved.	0
		0x3	RLS. 1 - Receive Line Status .	
		0x2	RDA. 2a - Receive Data Available.	
		0x6	CTI. 2b - Character Time-out Indicator.	
		0x1	THRE. 3 - THRE Interrupt.	
		0x0	Modem status. 4 - Modem status	
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFOEN		These bits are equivalent to FCR[0].	0
8	ABEOINT		End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.	0
9	ABTOINT		Auto-baud time-out interrupt. True if auto-baud has timed out and interrupt is enabled.	0
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

If the IntStatus bit is one and no interrupt is pending and the IntId bits will be zero. If the IntStatus is 0, a non auto-baud interrupt is pending in which case the IntId bits identify the type of interrupt and handling as described in Table 157. Given the status of IIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The IIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The USART0 RLS interrupt (IIR[3:1] = 011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the USART0 RX input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The USART0 Rx error condition that set the interrupt can be observed via LSR[4:1]. The interrupt is cleared upon a LSR read.

The USART0 RDA interrupt (IIR[3:1] = 010) shares the second level priority with the CTI interrupt (IIR[3:1] = 110). The RDA is activated when the USART0 Rx FIFO reaches the trigger level defined in FCR7:6 and is reset when the USART0 Rx FIFO depth falls below the trigger level. When the RDA interrupt goes active, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (IIR[3:1] = 110) is a second level interrupt and is set when the USART0 Rx FIFO contains at least one character and no USART0 Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any USART0 Rx FIFO activity (read or write of USART0 RSR) will clear the interrupt. This interrupt is intended to flush the USART0 RBR after a message has been received that is not a multiple of the trigger level size. For example, if a 105 character message was to be sent and the trigger level was 10 characters, the CPU would receive 10 RDA interrupts resulting in the transfer of 100 characters and 1 to 5 CTI interrupts (depending on the service routine) resulting in the transfer of the remaining 5 characters.

Table 157. USART0 Interrupt Handling

IIR[3:0] value[1]	Priority	Interrupt type	Interrupt source	Interrupt reset
0001	-	None	None	-
0110	Highest	RX Line Status / Error	OE[2] or PE[2] or FE[2] or BI[2]	LSR Read[2]
0100	Second	RX Data Available	Rx data available or trigger level reached in FIFO (FCR0=1)	RBR Read[3] or USART0 FIFO drops below trigger level
1100	Second	Character Time-out indication	Minimum of one character in the RX FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what the trigger level is set at (3.5 to 4.5 character times). The exact time will be: [(word length) × 7 - 2] × 8 + [(trigger level - number of characters) × 8 + 1] RCLKs	RBR Read[3]
0010	Third	THRE	THRE[2]	IIR Read[4] (if source of interrupt) or THR write
0000	Fourth	Modem Status	CTS, DSR, RI, or DCD.	MSR Read

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

[2] For details see Section 11.6.9 “USART0 Line Status Register (Read-Only)”

[3] For details see Section 11.6.1 “USART0 Receiver Buffer Register (when DLAB = 0, Read Only)”

[4] For details see Section 11.6.5 “USART0 Interrupt Identification Register (Read Only)” and Section 11.6.2 “USART0 Transmitter Holding Register (when DLAB = 0, Write Only)”

The USART0 THRE interrupt (IIR[3:1] = 001) is a third level interrupt and is activated when the USART0 THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the USART0 THR FIFO a chance to fill up with data to eliminate many THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever THRE = 1 and there have not been at least two characters in the THR at one time since the last THRE = 1 event. This delay is provided to give the CPU time to write data to THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the USART0 THR FIFO has held two or more characters at one time and currently, the THR is empty. The THRE interrupt is reset when a THR write occurs or a read of the IIR occurs and the THRE is the highest interrupt (IIR[3:1] = 001).

The modem status interrupt (IIR3:1 = 000) is the lowest priority USART0 interrupt and is activated whenever there is a state change on the CTS, DCD, or DSR or a trailing edge on the RI pin. The source of the modem interrupt can be read in MSR3:0. Reading the MSR clears the modem interrupt.

11.6.6 USART0 FIFO Control Register (Write Only)

The FCR controls the operation of the USART0 RX and TX FIFOs.

Table 158. USART0 FIFO Control Register Write only (FCR, address 0x4000 8008) bit description

Bit	Symbol	Value	Description	Reset value
0	FIFOEN		FIFO enable	0
		0	Disabled. USART0 FIFOs are disabled. Must not be used in the application.	
		1	Enabled. Active high enable for both USART0 Rx and TX FIFOs and FCR[7:1] access. This bit must be set for proper USART0 operation. Any transition on this bit will automatically clear the USART0 FIFOs.	
1	RXFIFORES		RX FIFO Reset	0
		0	No effect. No impact on either of USART0 FIFOs.	
		1	Clear. Writing a logic 1 to FCR[1] will clear all bytes in USART0 Rx FIFO, reset the pointer logic. This bit is self-clearing.	
2	TXFIFORES		TX FIFO Reset	0
		0	No effect. No impact on either of USART0 FIFOs.	
		1	Clear. Writing a logic 1 to FCR[2] will clear all bytes in USART0 TX FIFO, reset the pointer logic. This bit is self-clearing.	
3	DMAMODE		DMA Mode Select. When the FIFO enable bit (bit 0 of this register) is set, this bit selects the DMA mode.	0
		0	Disabled. DMA mode disabled.	
		1	Enable. DMA mode enabled.	
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	RXTL		RX Trigger Level. These two bits determine how many USART0 FIFO characters must be received by the FIFO before an interrupt is activated.	0
		0x0	Trigger level 0. (1 character or 0x01).	
		0x1	Trigger level 1. (4 characters or 0x04).	
		0x2	Trigger level 2. (8 characters or 0x08).	
		0x3	Trigger level 3. (14 characters or 0x0E).	
31:8	-	-	Reserved	-

11.6.7 USART0 Line Control Register

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

Table 159. USART0 Line Control Register (LCR, address 0x4000 800C) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	WLS		Word Length Select	0
		0x0	5 bit. 5-bit character length.	
		0x1	6 bit. 6-bit character length.	
		0x2	7 bit. 7-bit character length.	
		0x3	8 bit. 8-bit character length.	
2	SBS		Stop Bit Select	0
		0	1 stop bit.	
		1	2 stop bits. (1.5 if LCR[1:0]=00).	
3	PE		Parity Enable	0
		0	Disable. Disable parity generation and checking.	
		1	Enable. Enable parity generation and checking.	
5:4	PS		Parity Select	0
		0x0	Odd parity. Number of 1s in the transmitted character and the attached parity bit will be odd.	
		0x1	Even Parity. Number of 1s in the transmitted character and the attached parity bit will be even.	
		0x2	Forced 1 stick parity.	
		0x3	Forced 0 stick parity.	
6	BC		Break Control	0
		0	Disable. Disable break transmission.	
		1	Enable. Enable break transmission. Output pin USART0 TXD is forced to logic 0 when LCR[6] is active high.	
7	DLAB		Divisor Latch Access Bit	0
		0	Disable. Disable access to Divisor Latches.	
		1	Enable. Enable access to Divisor Latches.	
31:8	-	-	Reserved	-

11.6.8 USART0 Modem Control Register

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

Table 160. USART0 Modem Control Register (MCR, address 0x4000 8010) bit description

Bit	Symbol	Value	Description	Reset value
0	DTRCTRL		Source for modem output pin $\overline{\text{DTR}}$. This bit reads as 0 when modem loopback mode is active.	0
1	RTSCTRL		Source for modem output pin $\overline{\text{RTS}}$. This bit reads as 0 when modem loopback mode is active.	0
3:2	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

Table 160. USART0 Modem Control Register (MCR, address 0x4000 8010) bit description

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

11.6.9 USART0 Line Status Register (Read-Only)

The LSR is a read-only register that provides status information on the USART0 TX and RX blocks.

Table 161. USART0 Line Status Register Read only (LSR, address 0x4000 8014) bit description

Bit	Symbol	Value	Description	Reset Value
0	RDR		Receiver Data Ready:LSR[0] is set when the RBR holds an unread character and is cleared when the USART0 RBR FIFO is empty.	0
		0	Empty. RBR is empty.	
		1	Filled. RBR contains valid data.	
1	OE		Overrun Error. The overrun error condition is set as soon as it occurs. A LSR read clears LSR[1]. LSR[1] is set when USART0 RSR has a new character assembled and the USART0 RBR FIFO is full. In this case, the USART0 RBR FIFO will not be overwritten and the character in the USART0 RSR will be lost.	0
		0	Inactive. Overrun error status is inactive.	
		1	Active. Overrun error status is active.	

Table 161. USART0 Line Status Register Read only (LSR, address 0x4000 8014) bit description *…continued*

Bit	Symbol	Value	Description	Reset Value
2	PE		Parity Error. When the parity bit of a received character is in the wrong state, a parity error occurs. A LSR read clears LSR[2]. Time of parity error detection is dependent on FCR[0]. **Note:** A parity error is associated with the character at the top of the USART0 RBR FIFO.	0
		0	Inactive. Parity error status is inactive.	
		1	Active. Parity error status is active.	
3	FE		Framing Error. When the stop bit of a received character is a logic 0, a framing error occurs. A LSR read clears LSR[3]. The time of the framing error detection is dependent on FCR0. Upon detection of a framing error, the RX will attempt to re-synchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error. **Note:** A framing error is associated with the character at the top of the USART0 RBR FIFO.	0
		0	Inactive. Framing error status is inactive.	
		1	Active. Framing error status is active.	
4	BI		Break Interrupt. When RXD1 is held in the spacing state (all zeros) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RXD1 goes to marking state (all ones). A LSR read clears this status bit. The time of break detection is dependent on FCR[0]. **Note:** The break interrupt is associated with the character at the top of the USART0 RBR FIFO.	0
		0	Inactive. Break interrupt status is inactive.	
		1	Active. Break interrupt status is active.	
5	THRE		Transmitter Holding Register Empty. THRE is set immediately upon detection of an empty USART0 THR and is cleared on a THR write.	1
		0	Data. THR contains valid data.	
		1	Empty. THR is empty.	
6	TEMT		Transmitter Empty. TEMT is set when both THR and TSR are empty; TEMT is cleared when either the TSR or the THR contain valid data.	1
		0	Data. THR and/or the TSR contains valid data.	
		1	Empty. THR and the TSR are empty.	

Table 161. USART0 Line Status Register Read only (LSR, address 0x4000 8014) bit description *…continued*

Bit	Symbol	Value	Description	Reset Value
7	RXFE		Error in RX FIFO. LSR[7] is set when a character with a RX error such as framing error, parity error or break interrupt, is loaded into the RBR. This bit is cleared when the LSR register is read and there are no subsequent errors in the USART0 FIFO.	0
		0	Empty. RBR contains no USART0 RX errors or FCR[0]=0.	
		1	Error. USART0 RBR contains at least one USART0 RX error.	
8	TXERR		Tx Error. In smart card T=0 operation, this bit is set when the smart card has NACKed a transmitted character, one more than the number of times indicated by the TXRETRY field.	0
31:9	-	-	Reserved	-

11.6.10 USART0 Modem Status Register

The MSR is a read-only register that provides status information on USART0 input signals. Bit 0 is cleared when (after) this register is read.

Table 162: USART0 Modem Status Register (MSR, address 0x4000 8018) bit description

Bit	Symbol	Value	Description	Reset value
0	DCTS		Delta CTS. Set upon state change of input CTS. Cleared on an MSR read.	0
		0	No change. No change detected on modem input, CTS.	
		1	Changed. State change detected on modem input, CTS.	
1	DDSR		Delta DSR. Set upon state change of input DSR. Cleared on an MSR read.	0
		0	No change. No change detected on modem input, DSR.	
		1	Changed. State change detected on modem input, DSR.	
2	TERI		Trailing Edge RI. Set upon low to high transition of input RI. Cleared on an MSR read.	0
		0	No change. No change detected on modem input, RI.	
		1	Transition. Low-to-high transition detected on RI.	
3	DDCD		Delta DCD. Set upon state change of input DCD. Cleared on an MSR read.	0
		0	No change. No change detected on modem input, DCD.	
		1	Changed. State change detected on modem input, DCD.	
4	CTS	-	Clear To Send State. Complement of input signal CTS. This bit is connected to MCR[1] in modem loopback mode.	0
5	DSR	-	Data Set Ready State. Complement of input signal DSR. This bit is connected to MCR[0] in modem loopback mode.	0

Table 162: USART0 Modem Status Register (MSR, address 0x4000 8018) bit description

Bit	Symbol	Value	Description	Reset value
6	RI	-	Ring Indicator State. Complement of input RI. This bit is connected to MCR[2] in modem loopback mode.	0
7	DCD	-	Data Carrier Detect State. Complement of input DCD. This bit is connected to MCR[3] in modem loopback mode.	0
31:8	-	-	Reserved, the value read from a reserved bit is not defined.	NA

11.6.11 USART0 Scratch Pad Register

The SCR has no effect on the USART0 operation. This register can be written and/or read at user’s discretion. There is no provision in the interrupt interface that would indicate to the host that a read or write of the SCR has occurred.

Table 163. USART0 Scratch Pad Register (SCR, address 0x4000 801C) bit description

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

11.6.12 USART0 Auto-baud Control Register

The USART0 Auto-baud Control Register (ACR) controls the process of measuring the incoming clock/data rate for baud rate generation, and can be read and written at the user’s discretion.

Table 164. Auto-baud Control Register (ACR, address 0x4000 8020) bit description

Bit	Symbol	Value	Description	Reset value
0	START		This bit is automatically cleared after auto-baud completion.	0
		0	Stop. Auto-baud stop (auto-baud is not running).	
		1	Start. Auto-baud start (auto-baud is running). Auto-baud run bit. This bit is automatically cleared after auto-baud completion.	
1	MODE		Auto-baud mode select bit.	0
		0	Mode 0.	
		1	Mode 1.	
2	AUTORESTART		Start mode	0
		0	No restart.	
		1	Time-out restart. Restart in case of time-out (counter restarts at next USART0 Rx falling edge)	
7:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
8	ABEOINTCLR		End of auto-baud interrupt clear bit (write only accessible).	0
		0	No effect. Writing a 0 has no impact.	
		1	Clear. Writing a 1 will clear the corresponding interrupt in the IIR.	

Table 164. Auto-baud Control Register (ACR, address 0x4000 8020) bit description

Bit	Symbol	Value	Description	Reset value
9	ABTOINTCLR		Auto-baud time-out interrupt clear bit (write only accessible).	0
		0	No effect. Writing a 0 has no impact.	
		1	Clear. Writing a 1 will clear the corresponding interrupt in the IIR.	
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

11.6.13 IrDA Control Register

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

Table 165: IrDA Control Register (ICR, 0x4000 8024) bit description

Bit	Symbol	Value	Description	Reset value
0	IRDAEN		IrDA mode enable	0
		0	Disabled. IrDA mode is disabled.	
		1	Enabled. IrDA mode is enabled.	
1	IRDAINV		Serial input inverter	0
		0	Not inverted. The serial input is not inverted.	
		1	Inverted. The serial input is inverted. This has no effect on the serial output.	
2	FIXPULSEEN		IrDA fixed pulse width mode.	0
		0	Disabled. IrDA fixed pulse width mode disabled.	
		1	Enabled. IrDA fixed pulse width mode enabled.	
5:3	PULSEDIV		Configures the pulse width when FixPulseEn = 1.	0
		0x0	3 DIV 16. 3 / (16 × baud rate)	
		0x1	2 x T_{PCLK}.	
		0x2	4 x T_{PCLK}.	
		0x3	8 x T_{PCLK}.	
		0x4	16 x $T_{PCLK.}$	
		0x5	32 x T_{PCLK}.	
		0x6	64 x T_{PCLK}.	
		0x7	128 x T_{PCLK}.	
31:6	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

The PulseDiv bits in the ICR are used to select the pulse width when the fixed pulse width mode is used in IrDA mode (IrDAEn = 1 and FixPulseEn = 1). The value of these bits should be set so that the resulting pulse width is at least 1.63 μs. Table 166 shows the possible pulse widths.

Table 166: IrDA Pulse Width

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

11.6.14 USART0 Fractional Divider Register

The USART0 Fractional Divider Register (FDR) controls the clock pre-scaler for the baud rate generation and can be read and written at the user’s discretion. This pre-scaler takes the APB clock and generates an output clock according to the specified fractional requirements.

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

Table 167. USART0 Fractional Divider Register (FDR, address 0x4000 8028) bit description

Bit	Function	Description	Reset value
3:0	DIVADDVAL	Baud rate generation pre-scaler divisor value. If this field is 0, fractional baud rate generator will not impact the USART0 baud rate.	0
7:4	MULVAL	Baud rate pre-scaler multiplier value. This field must be greater or equal 1 for USART0 to operate properly, regardless of whether the fractional baud rate generator is used or not.	1
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

This register controls the clock pre-scaler for the baud rate generation. The reset value of the register keeps the fractional capabilities of USART0 disabled making sure that USART0 is fully software and hardware compatible with USARTs not equipped with this feature.

The USART0 baud rate can be calculated as:

$$UART_{baudrate} = \frac{PCLK}{16 \times (256 \times DLM + DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)} \tag{2}$$

Where UART_PCLK is the peripheral clock, DLM and DLL are the standard USART0 baud rate divider registers, and DIVADDVAL and MULVAL are USART0 fractional baud rate generator specific parameters.

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

1. 1 ≤ MULVAL ≤ 15

2. $0 \leq$ DIVADDVAL ≤ 14
3. DIVADDVAL< MULVAL

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

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

11.6.15 USART0 Oversampling Register

In most applications, the USART0 samples received data 16 times in each nominal bit time, and sends bits that are 16 input clocks wide. This register allows software to control the ratio between the input clock and bit clock. This is required for smart card mode, and provides an alternative to fractional division for other modes.

Table 168. USART0 Oversampling Register (OSR, address 0x4000 802C) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3:1	OSFRAC	Fractional part of the oversampling ratio, in units of 1/8th of an input clock period. (001 = 0.125, ..., 111 = 0.875)	0
7:4	OSINT	Integer part of the oversampling ratio, minus 1. The reset values equate to the normal operating mode of 16 input clocks per bit time.	0xF
14:8	FDINT	In Smart Card mode, these bits act as a more-significant extension of the OSint field, allowing an oversampling ratio up to 2048 as required by ISO7816-3. In Smart Card mode, bits 14:4 should initially be set to 371, yielding an oversampling ratio of 372.	0
31:15	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Example: For a baud rate of 3.25 Mbps with a 24 MHz USART0 clock frequency, the ideal oversampling ratio is 24/3.25 or 7.3846. Setting OSINT to 0110 for 7 clocks/bit and OSFrac to 011 for 0.375 clocks/bit, results in an oversampling ratio of 7.375.

In Smart card mode, OSInt is extended by FDINT. This extends the possible oversampling to 2048, as required to support ISO 7816-3. Note that this value can be exceeded when D<0, but this is not supported by the USART0. When Smart card mode is enabled, the initial value of OSINT and FDINT should be programmed as “00101110011” (372 minus one).

11.6.16 USART0 Transmit Enable Register

In addition to being equipped with full hardware flow control (auto-cts and auto-rts mechanisms described above), TER enables implementation of software flow control. When TxEn = 1, the USART0 transmitter will keep sending data as long as they are available. As soon as TxEn becomes 0, USART0 transmission will stop.

Although Table 169 describes how to use TxEn bit in order to achieve hardware flow control, it is strongly suggested to let the USART0 hardware implemented auto flow control features take care of this and limit the scope of TxEn to software flow control.

Table 169 describes how to use TXEn bit in order to achieve software flow control.

Table 169. USART0 Transmit Enable Register (TER, address 0x4000 8030) bit description

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

11.6.17 UART Half-duplex enable register

Remark: The HDEN register should be disabled when in smart card mode or IrDA mode (smart card and IrDA by default run in half-duplex mode).

After reset the USART0 will be in full-duplex mode, meaning that both TX and RX work independently. After setting the HDEN bit, the USART0 will be in half-duplex mode. In this mode, the USART0 ensures that the receiver is locked when idle, or will enter a locked state after having received a complete ongoing character reception. Line conflicts must be handled in software. The behavior of the USART0 is unpredictable when data is presented for reception while data is being transmitted.

For this reason, the value of the HDEN register should not be modified while sending or receiving data, or data may be lost or corrupted.

Table 170. USART0 Half duplex enable register (HDEN, address 0x4000 8040) bit description

Bit	Symbol	Value	Description	Reset value
0	HDEN		Half-duplex mode enable	0
		0	Disable. Disable half-duplex mode.	
		1	Enable. Enable half-duplex mode.	
31:1	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

11.6.18 Smart Card Interface Control register

This register allows the USART0 to be used in asynchronous smart card applications.

Table 171. Smart Card Interface Control register (SCICTRL, address 0x4000 8048) bit description

Bit	Symbol	Value	Description	Reset value
0	SCIEN		Smart Card Interface Enable.	0
		0	Disabled. Smart card interface disabled.	
		1	Enabled. Asynchronous half duplex smart card interface is enabled.	
1	NACKDIS		NACK response disable. Only applicable in T=0.	0
		0	Enabled. A NACK response is enabled.	
		1	Inhibited. A NACK response is inhibited.	
2	PROTSEL		Protocol selection as defined in the ISO7816-3 standard.	0
		0	T = 0.	
		1	T = 1.	
4:3	-	-	Reserved.	-
7:5	TXRETRY	-	When the protocol selection T bit (above) is 0, the field controls the maximum number of retransmissions that the USART0 will attempt if the remote device signals NACK. When NACK has occurred this number of times plus one, the Tx Error bit in the LSR is set, an interrupt is requested if enabled, and the USART0 is locked until the FIFO is cleared.	-
15:8	XTRAGUARD	-	When the protocol selection T bit (above) is 0, this field indicates the number of bit times (ETUs) by which the guard time after a character transmitted by the USART0 should exceed the nominal 2 bit times. 0xFF in this field may indicate that there is just a single bit after a character and 11 bit times/character	
31:16	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

11.6.19 USART0 RS485 Control register

The RS485CTRL register controls the configuration of the USART0 in RS-485/EIA-485 mode.

Table 172. USART0 RS485 Control register (RS485CTRL, address 0x4000 804C) bit description

Bit	Symbol	Value	Description	Reset value
0	NMMEN		NMM enable.	0
		0	Disabled. RS-485/EIA-485 Normal Multidrop Mode (NMM) is disabled.	
		1	Enabled. RS-485/EIA-485 Normal Multidrop Mode (NMM) is enabled. In this mode, an address is detected when a received byte causes the USART0 to set the parity error and generate an interrupt.	

Table 172. USART0 RS485 Control register (RS485CTRL, address 0x4000 804C) bit description *…continued*

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

11.6.20 USART0 RS-485 Address Match register

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

Table 173. USART0 RS-485 Address Match register (RS485ADRMATCH, address 0x4000 8050) bit description

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

11.6.21 USART0 RS-485 Delay value register

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

Table 174. USART0 RS-485 Delay value register (RS485DLY, address 0x4000 8054) bit description

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

11.6.22 USART0 Synchronous mode control register

SYNCCTRL register controls the synchronous mode. When this mode is in effect, the USART0 generates or receives a bit clock on the SCLK pin and applies it to the transmit and receive shift registers. Synchronous mode should not be used with smart card mode.

Table 175. USART0 Synchronous mode control register (SYNCCTRL, address 0x4000 8058) bit description

Bit	Symbol	Value	Description	Reset value
0	SYNC		Enables synchronous mode.	0
		0	Disabled.	
		1	Enabled.	
1	CSRC		Clock source select.	0
		0	Slave. Synchronous slave mode (SCLK in)	
		1	Master. Synchronous master mode (SCLK out)	
2	FES		Falling edge sampling.	0
		0	Rising edge. RXD is sampled on the rising edge of SCLK	
		1	Falling edge. RXD is sampled on the falling edge of SCLK	
3	TSBYPASS		Transmit synchronization bypass in synchronous slave mode.	0
		0	Synchronous. The input clock is synchronized prior to being used in clock edge detection logic.	
		1	Asynchronous. The input clock is not synchronized prior to being used in clock edge detection logic. This allows for a high er input clock rate at the expense of potential metastability.	
4	CSCEN		Continuous master clock enable (used only when CSRC is 1)	0
		0	SCLK cycles only when characters are being sent on TXD.	
		1	SCLK runs continuously (characters can be received on RXD independently from transmission on TxD).	
5	SSDIS		Start/stop bits	0
		0	Send. Send start and stop bits as in other modes.	
		1	Not send. Do not send start/stop bits.	

Table 175. USART0 Synchronous mode control register (SYNCCTRL, address 0x4000 8058) bit description

Bit	Symbol	Value	Description	Reset value
6	CCCLR		Continuous clock clear	0
		0	Software control. CSCEN is under software control.	
		1	Hardware control. Hardware clears CSCEN after each character is received.	
31:7	-		Reserved. The value read from a reserved bit is not defined.	NA

After reset, synchronous mode is disabled. Synchronous mode is enabled by setting the SYNC bit. When SYNC is 1, the USART0 operates as follows:

1. The CSRC bit controls whether the USART0 sends (master mode) or receives (slave mode) a serial bit clock on the SCLK pin.
2. When CSRC is 1 selecting master mode, the CSCEN bit selects whether the USART0 produces clocks on SCLK continuously (CSCEN=1) or only when transmit data is being sent on TxD (CSCEN=0).
3. The SSDIS bit controls whether start and stop bits are used. When SSDIS is 0, the USART0 sends and samples for start and stop bits as in other modes. When SSDIS is 1, the USART0 neither sends nor samples for start or stop bits, and each falling edge on SCLK samples a data bit on RxD into the receive shift register, as well as shifting the transmit shift register.

The rest of this section provides further details of operation when SYNC is 1.

Data changes on TxD from falling edges on SCLK. When SSDIS is 0, the FES bit controls whether the USART0 samples serial data on RxD on rising edges or falling edges on SCLK. When SSDIS is 1, the USART0 ignores FES and always samples RxD on falling edges on SCLK.

The combination SYNC=1, CSRC=1, CSCEN=1, and SSDIS=1 is a difficult operating mode, because SCLK applies to both directions of data flow and there is no defined mechanism to signal the receivers when valid data is present on TxD or RxD.

Lacking such a mechanism, SSDIS=1 can be used with CSCEN=0 or CSRC=0 in a mode similar to the SPI protocol, in which characters are (at least conceptually) “exchanged” between the USART0 and remote device for each set of 8 clock cycles on SCLK. Such operation can be called full-duplex, but the same hardware mode can be used in a half-duplex way under control of a higher-layer protocol, in which the source of SCLK toggles it in groups of N cycles whenever data is to be sent in either direction. (N being the number of bits/character.)

When the USART0 is the clock source (CSRC=1), such half-duplex operation can lead to the rather artificial-seeming requirement of writing a dummy character to the Transmitter Holding Register in order to generate 8 clocks so that a character can be received. The CCCLR bit provides a more natural way of programming half-duplex reception. When the higher-layer protocol dictates that the USART0 should receive a character, software should write the SYNCCTRL register with CSCEN=1 and CCCLR=1. After the USART0 has sent N clock cycles and thus received a character, it clears the CSCEN bit. If more characters need to be received thereafter, software can repeat setting CSCEN and CCCLR.

Aside from half-duplex operation, the primary use of CSCEN=1 is with SSDIS=0, so that start bits indicate the transmission of each character in each direction.

11.7 Functional description

11.7.1 DMA Operation

The user can optionally operate the UART transmit and/or receive using DMA. The DMA mode is determined by the DMA Mode Select bit in the FCR register. Note that for DMA operation as for any operation of the USART, the FIFOs must be enabled via the FIFO Enable bit in the FCR register.

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

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

11.7.2 Auto-flow control

If auto-RTS mode is enabled, the USART0‘s receiver FIFO hardware controls the $\overline{\text{RTS}}$ output of the USART0. If the auto-CTS mode is enabled, the USART0‘s transmitter will only start sending if the $\overline{\text{CTS}}$ pin is low.

11.7.2.1 Auto-RTS

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

When the receiver FIFO level reaches the programmed trigger level, $\overline{\text{RTS}}$ is deasserted (to a high value). It is possible that the sending USART0 sends an additional byte after the trigger level is reached (assuming the sending USART0 has another byte to send) because it might not recognize the deassertion of $\overline{\text{RTS}}$ until after it has begun sending the additional byte. $\overline{\text{RTS}}$ is automatically reasserted (to a low value) once the receiver FIFO has reached the previous trigger level. The reassertion of $\overline{\text{RTS}}$ signals the sending USART0 to continue transmitting data.

If Auto-RTS mode is disabled, the RTSen bit controls the $\overline{\text{RTS}}$ output of the USART0. If Auto-RTS mode is enabled, hardware controls the $\overline{\text{RTS}}$ output, and the actual value of $\overline{\text{RTS}}$ will be copied in the RTS Control bit of the USART0. As long as Auto-RTS is enabled, the value of the RTS Control bit is read-only for software.

Example: Suppose the USART0 operating in type ‘550 mode has the trigger level in FCR set to 0x2, then, if Auto-RTS is enabled, the USART0 will deassert the $\overline{\text{RTS}}$ output as soon as the receive FIFO contains 8 bytes (Table 158 on page 169). The $\overline{\text{RTS}}$ output will be reasserted as soon as the receive FIFO hits the previous trigger level: 4 bytes.

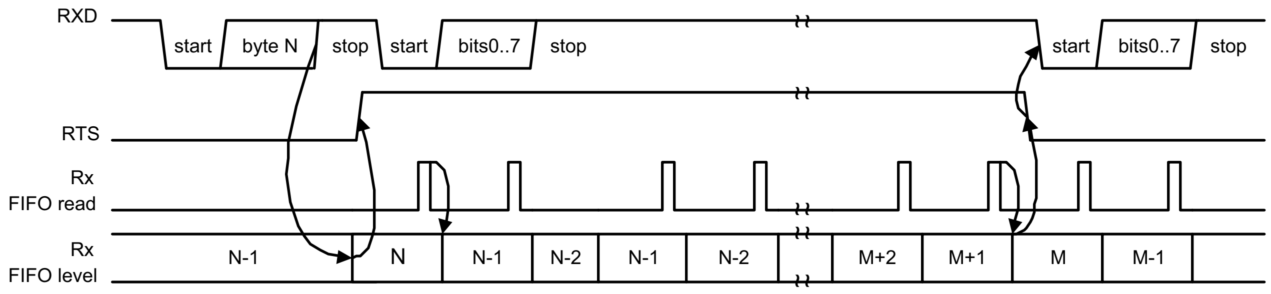


Fig 19. Auto-RTS Functional Timing

11.7.2.2 Auto-CTS

The Auto-CTS function is enabled by setting the CTSen bit. If Auto-CTS is enabled, the transmitter circuitry checks the CTS input before sending the next data byte. When CTS is active (low), the transmitter sends the next byte. To stop the transmitter from sending the following byte, CTS must be released before the middle of the last stop bit that is currently being sent. In Auto-CTS mode, a change of the CTS signal does not trigger a modem status interrupt unless the CTS Interrupt Enable bit is set, but the Delta CTS bit in the MSR will be set. Table 176 lists the conditions for generating a Modem Status interrupt.

Table 176. Modem status interrupt generation

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

The auto-CTS function typically eliminates the need for CTS interrupts. When flow control is enabled, a CTS state change does not trigger host interrupts because the device automatically controls its own transmitter. Without Auto-CTS, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result. Figure 20 illustrates the Auto-CTS functional timing.

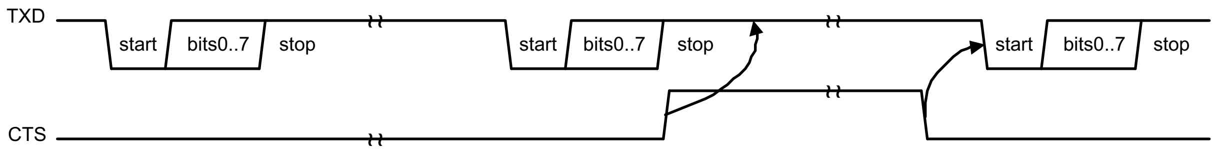


Fig 20. Auto-CTS Functional Timing

During transmission of the second character the $\overline{\text{CTS}}$ signal is negated. The third character is not sent thereafter. The USART0 maintains 1 on TXD as long as $\overline{\text{CTS}}$ is negated (high). As soon as $\overline{\text{CTS}}$ is asserted, transmission resumes and a start bit is sent followed by the data bits of the next character.

11.7.3 Auto-baud

The USART0 auto-baud function can be used to measure the incoming baud rate based on the “AT” protocol (Hayes command). If enabled the auto-baud feature will measure the bit time of the receive data stream and set the divisor latch registers DLM and DLL accordingly.

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

Two auto-baud measuring modes are available which can be selected by the ACR Mode bit. In Mode 0 the baud rate is measured on two subsequent falling edges of the USART0 Rx pin (the falling edge of the start bit and the falling edge of the least significant bit). In Mode 1 the baud rate is measured between the falling edge and the subsequent rising edge of the USART0 Rx pin (the length of the start bit).

The ACR AutoRestart bit can be used to automatically restart baud rate measurement if a time-out occurs (the rate measurement counter overflows). If this bit is set, the rate measurement will restart at the next falling edge of the USART0 Rx pin.

The auto-baud function can generate two interrupts.

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

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

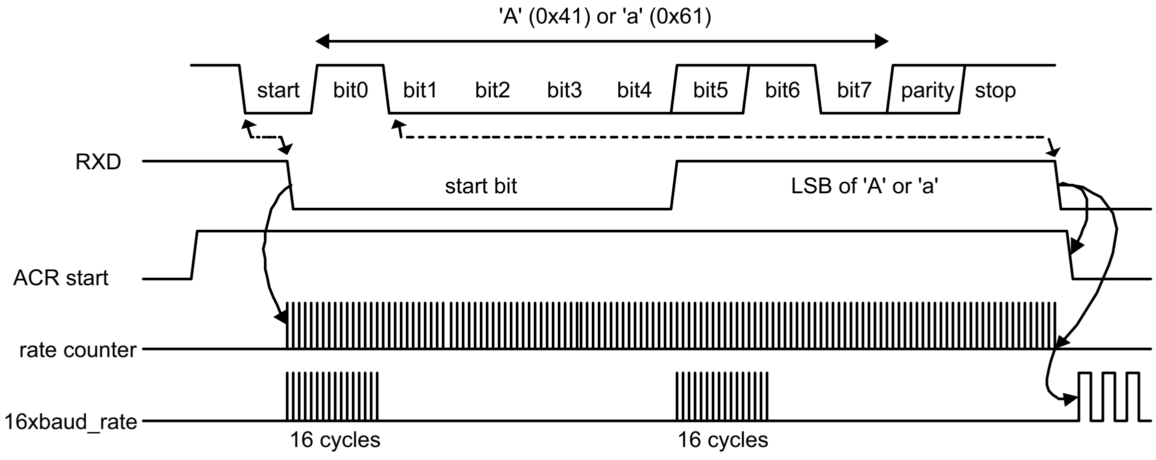
The fractional baud rate generator must be disabled (DIVADDVAL = 0) during auto-baud. Also, when auto-baud is used, any write to DLM and DLL registers should be done before ACR register write. The minimum and the maximum baud rates supported by USART0 are a function of USART0_PCLK and the number of data bits, stop bits and parity bits.

$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \le UART_{baudrate} \le \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax \tag{3}$$

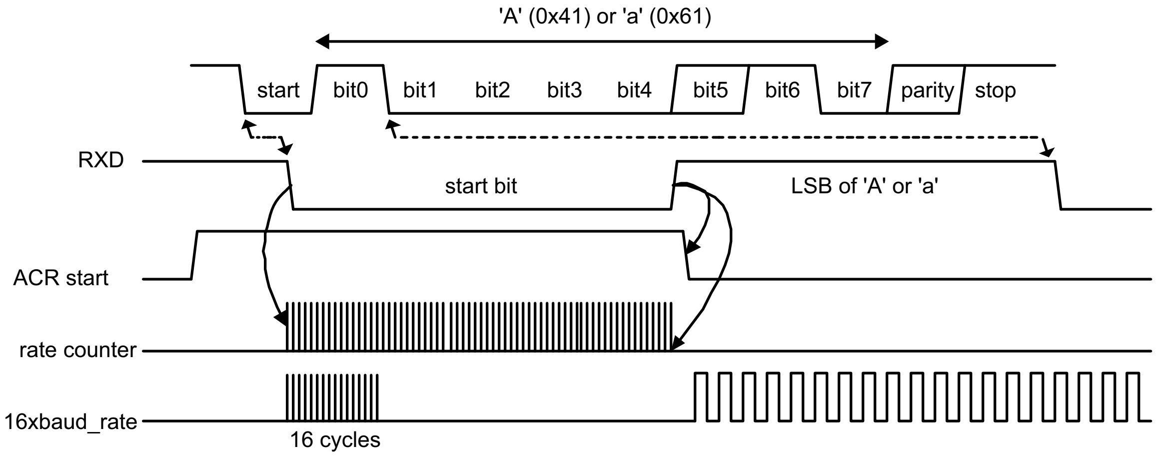
11.7.4 Auto-baud modes

When the software is expecting an “AT” command, it configures the USART0 with the expected character format and sets the ACR Start bit. The initial values in the divisor latches DLM and DLM don‘t care. Because of the “A” or “a” ASCII coding (“A” = 0x41, “a” = 0x61), the USART0 Rx pin sensed start bit and the LSB of the expected character are delimited by two falling edges. When the ACR Start bit is set, the auto-baud protocol will execute the following phases:

1. On ACR Start bit setting, the baud rate measurement counter is reset and the USART0 RSR is reset. The RSR baud rate is switched to the highest rate.
2. A falling edge on USART0 Rx pin triggers the beginning of the start bit. The rate measuring counter will start counting UART_PCLK cycles.
3. During the receipt of the start bit, 16 pulses are generated on the RSR baud input with the frequency of the USART0 input clock, guaranteeing the start bit is stored in the RSR.
4. During the receipt of the start bit (and the character LSB for Mode = 0), the rate counter will continue incrementing with the pre-scaled USART0 input clock (UART_PCLK).
5. If Mode = 0, the rate counter will stop on next falling edge of the USART0 Rx pin. If Mode = 1, the rate counter will stop on the next rising edge of the USART0 Rx pin.
6. The rate counter is loaded into DLM/DLL and the baud rate will be switched to normal operation. After setting the DLM/DLL, the end of auto-baud interrupt IIR ABEOInt will be set, if enabled. The RSR will now continue receiving the remaining bits of the character.



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



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

Fig 21. Auto-baud a) mode 0 and b) mode 1 waveform

11.7.5 Baud rate calculation in asynchronous mode

The USART0 baud rate can be calculated as:

$$UART_{baudrate} = \frac{PCLK}{16 \times (256 \times DLM + DLL) \times \left(1 + \frac{DIVADDVAL}{MULVAL}\right)} \tag{4}$$

Where UART_PCLK is the peripheral clock (PCLK = main clock/USART0CLKDIV), DLM and DLL are the standard USART0 baud rate divider registers, and DIVADDVAL and MULVAL are USART0 fractional baud rate generator specific parameters.

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

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

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

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

The USART0 can operate with or without using the Fractional Divider. In real-life applications it is likely that the desired baud rate can be achieved using several different Fractional Divider settings. The following algorithm illustrates one way of finding a set of DLM, DLL, MULVAL, and DIVADDVAL values. Such a set of parameters yields a baud rate with a relative error of less than 1.1% from the desired one.

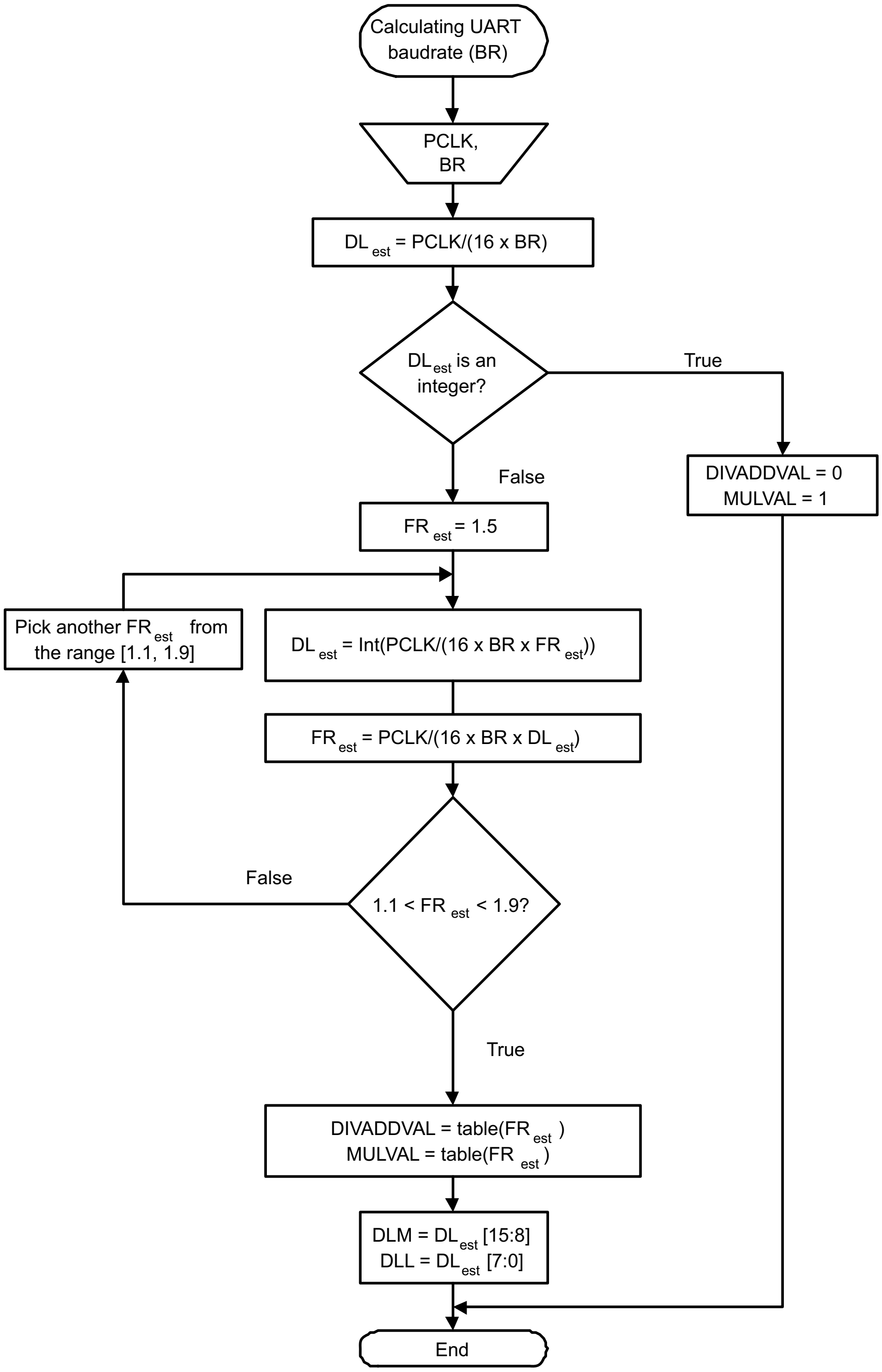


Fig 22. Algorithm for setting USART0 dividers

Table 177. Fractional Divider setting look-up table

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

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

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

11.7.5.2 Example 2: UART_PCLK = 12.0 MHz, BR = 115200

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

The closest value for FRest = 1.628 in the look-up Table 177 is FR = 1.625. It is equivalent to DIVADDVAL = 5 and MULVAL = 8.

Based on these findings, the suggested USART0 setup would be: DLM = 0, DLL = 4, DIVADDVAL = 5, and MULVAL = 8. According to Equation 2, the USART0's baud rate is 115384. This rate has a relative error of 0.16% from the originally specified 115200.

11.7.6 USART clock in synchronous mode

In synchronous master mode, the USART synchronous clock is determined as follows:

$$U0\_SCLK = \frac{main\ \text{clock}}{2 \times USART0CLKDIV \times (256 \times DLM + DLL) \times \left(1 + \frac{DIVADDVAL}{MULVAL}\right)} \tag{5}$$

DLM and DLL are the standard USART0 baud rate divider registers, and DIVADDVAL and MULVAL are USART0 fractional baud rate generator specific parameters. Setting DIVADDVAL = 0 disables the fractional baud rate generator.

11.7.7 RS-485/EIA-485 modes of operation

The RS-485/EIA-485 feature allows the USART0 to be configured as an addressable slave. The addressable slave is one of multiple slaves controlled by a single master.

The USART0 master transmitter will identify an address character by setting the parity (9th) bit to ‘1’. For data characters, the parity bit is set to ‘0’.

Each USART0 slave receiver can be assigned a unique address. The slave can be programmed to either manually or automatically reject data following an address which is not theirs.

RS-485/EIA-485 Normal Multidrop Mode

Setting the RS485CTRL bit 0 enables this mode. In this mode, an address is detected when a received byte causes the USART0 to set the parity error and generate an interrupt.

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

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

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

When both RS485CTRL register bits 0 (9-bit mode enable) and 2 (AAD mode enable) are set, the USART0 is in auto address detect mode.

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

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

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

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

RS-485/EIA-485 Auto Direction Control

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

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

Keep RS485CTRL bit 3 zero so that direction control, if enabled, will use the $\overline{\text{RTS}}$ pin.

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

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

RS485/EIA-485 driver delay time

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

RS485/EIA-485 output inversion

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

11.7.8 Smart card mode

Figure 23 shows a typical asynchronous smart card application.

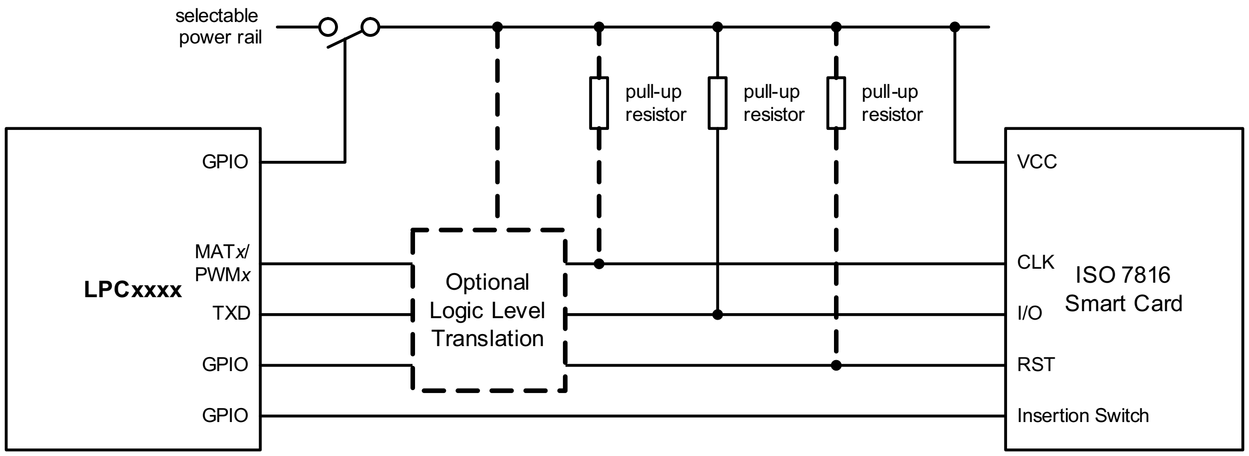


Fig 23. Typical smart card application

When the SCIEN bit in the SCICTRL register (Table 171) is set as described, the USART0 provides bidirectional serial data on the open-drain TXD pin. No RXD pin is used when SCIEN is 1. If a clock source is needed as an oscillator source into the Smart Card, a timer match or PWM output can be used in cases when a higher frequency clock is needed that is not synchronous with the data bit rate. The USART0 SCLK pin will output synchronously with the data and at the data bit rate and may not be adequate for most asynchronous cards. Software must use timers to implement character and block waiting times (no hardware support via trigger signals is provided on this part). GPIO pins can be used to control the smart card reset and power pins. Any power supplied to the card must be externally switched as card power supply requirements often exceed source currents possible on this part. As the specific application may accommodate any of the available ISO 7816 class A, B, or C power requirements, be aware of the logic level tolerances and requirements when communicating or powering cards that use different power rails than this part.

11.7.8.1 Smart card set-up procedure

A T = 0 protocol transfer consists of 8-bits of data, an even parity bit, and two guard bits that allow for the receiver of the particular transfer to flag parity errors through the NACK response (see Figure 24). Extra guard bits may be added according to card requirements. If no NACK is sent (provided the interface accepts them in SCICTRL), the next byte may be transmitted immediately after the last guard bit. If the NACK is sent, the transmitter will retry sending the byte until successfully received or until the SCICTRL retry limit has been met.

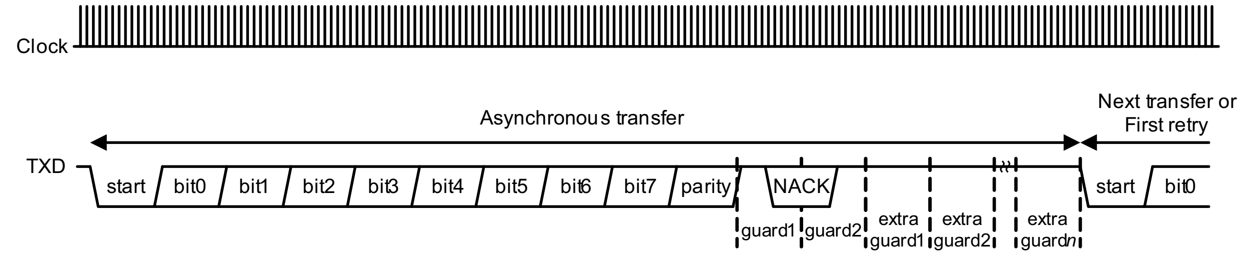


Fig 24. Smart card T = 0 waveform

The smart card must be set up with the following considerations:

- If necessary, program PRESETCTRL (Table 23) to release the USART0 peripheral reset.
- Program one IOCON register to enable a USART0 TXD function.
- If the smart card to be communicated with requires a clock, program one IOCON register for the USART0 SCLK function. The USART0 will use it as an output.
- Program USART0CLKDIV (Table 42) for an initial USART0 frequency of 3.58 MHz.
- Program the OSR (Section 11.6.15) for 372x oversampling.
- If necessary, program the DLM and DLL (Section 11.6.3) to 00 and 01 respectively, to pass the USART0 clock through without division.
- Program the LCR (Section 11.6.7) for 8-bit characters, parity enabled, even parity.
- Program the GPIO signals associated with the smart card so that (in this order):
  a. Reset is low.
  b. VCC is provided to the card (GPIO pins do not have the required 200 mA drive).
  c. VPP (if provided to the card) is at “idle” state.
- Program SCICTRL (Section 11.6.18) to enable the smart card feature with the desired options.
- Set up one or more timers to provide timing as needed for ISO 7816 startup.
- Program SYSAHBCLKCTRL (Table 40) to enable the USART0 clock.

Thereafter, software should monitor card insertion, handle activation, wait for answer to reset as described in ISO7816-3.

UM10732

Chapter 12: LPC11U6x/E6x USART1/2/3/4

Rev. 1.8 — 19 December 2016 **User manual**

12.1 How to read this chapter

USART1 and USART2 are available on all parts. USART3 is available on LPC11U68JBD100, LPC11E67JBD48, LPC11E66, and LPC11E68JBD64. USART4 is available on parts LPC11U68JBD100 and LPC11E68JBD100 only.

Remark: The USART1 to USART4 register maps and register functions are identical. The USART0 peripheral has a different register map and different register functions.

12.2 Features

- 7, 8, or 9 data bits and 1 or 2 stop bits
- Synchronous mode with master or slave operation. Includes data phase selection and continuous clock option.
- Multiprocessor/multidrop (9-bit) mode with software address compare.
- RS-485 transceiver output enable.
- Parity generation and checking: odd, even, or none.
- Software selectable oversampling from 5 to 16 clocks in asynchronous mode.
- One transmit and one receive data buffer.
- RTS/CTS for hardware signaling for automatic flow control. Software flow control can be performed using Delta CTS detect, Transmit Disable control, and any GPIO as an RTS output.
- Received data and status can optionally be read from a single register
- Break generation and detection.
- Receive data is 2 of 3 sample "voting". Status flag set when one sample differs.
- Built-in Baud Rate Generator with autobaud function.
- A fractional rate divider is shared among all USARTs.
- Interrupts available for Receiver Ready, Transmitter Ready, Receiver Idle, change in receiver break detect, Framing error, Parity error, Overrun, Underrun, Delta CTS detect, and receiver sample noise detected.
- Loopback mode for testing of data and flow control.
- Special operating mode allows operation at up to 9600 baud using the 32 kHz RTC oscillator (32, 768 kHz) as the UART clock. This mode can be used while the device is in Deep-sleep or power-down mode and can wake-up the device when a character is received.
- USARTn transmit and receive functions can operated with the system DMA controller.

12.3 Basic configuration

Remark: The on-chip USART API provides software routines to configure and use the USART. See Table 437.

Configure USART1/2/3/4 for receiving and transmitting data:

- In the SYSAHBCLKCTRL register, set bit 20 to 22 (Table 40) to enable the clock to the register interface. USART3 and USART4 use a common clock.
- Clear the USART1/2/3/4 peripheral resets using the PRESETCTRL register (Table 23).
- Enable or disable the USART1/2/3/4 interrupts in slots #11 and #12 in the NVIC. USART1 and USART4 interrupts are combined in slot #11. USART2 and USART3 interrupts are combined in slot #12. See Table 6.
- Configure the USART1/2/3/4 pin functions in the IOCON block. See Table 83.
- Configure the USART clock and baud rate. See Section 12.3.1.
- Send and receive lines are connected to DMA request lines. See Table 121.

Configure the USART1/2/3/4 to wake up the part from low power modes:

- Configure the USART to receive and transmit data in synchronous slave mode. See Section 12.3.2.

12.3.1 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 25):

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

   Table 44 “UART Fractional baud rate clock divider register (FRGCLKDIV, address 0x4004 80A0) bit description”
2. If a fractional value is needed to obtain a particular baud rate, program the fractional divider. The fractional divider value is the fraction of MULT/DIV. The MULT and DIV values are programmed in the FRGCTRL register. The DIV value must be programmed with the fixed value of 256.

   U_PCLK = FRGCLKDIV/(1+(MULT/DIV))

   The following rules apply for MULT and DIV:

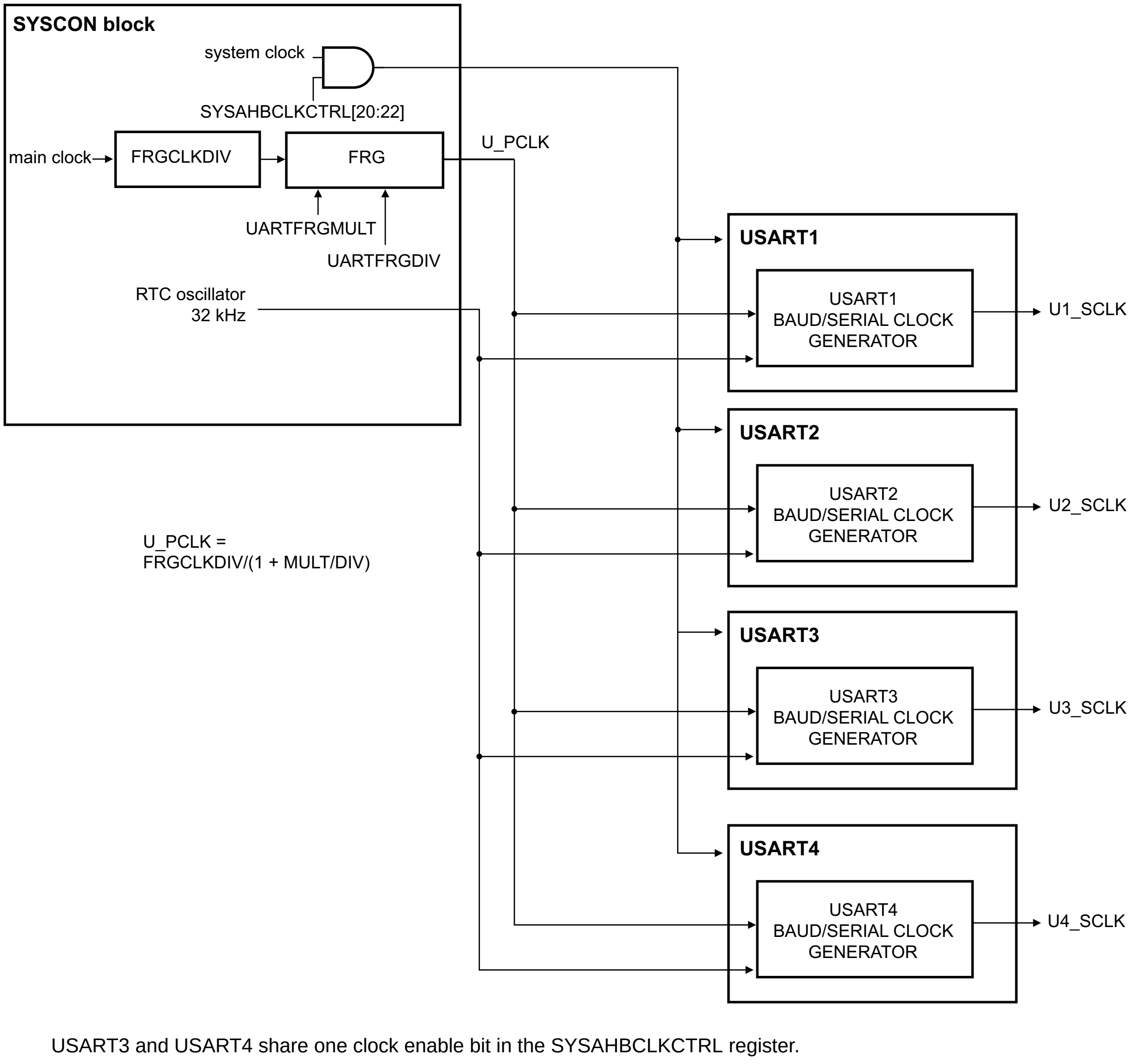
   - Always set DIV to 256 by programming the FRGCTRL register with the value of 0xFF.
   - Set the MULT to any value between 0 and 255.

   Table 51 “USART fractional generator divider value register (UARTFRGDIV, address 0x4004 80F0) bit description”

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

   Section 12.6.9 “USART Baud Rate Generator register”
4. In synchronous mode: The serial clock is Un_SCLK = U_PCLK/BRGVAL.

The USART can also be clocked by the 32 kHz RTC oscillator. Set the MODE32K bit to enable this 32 kHz mode. See also Section 12.7.1.4 “32 kHz mode”.



USART3 and USART4 share one clock enable bit in the SYSAHBCLKCTRL register.

Fig 25. USART clocking

For details on the clock configuration see:

Section 12.7.1 “Clocking and baud rates”

12.3.2 Configure the USART for wake-up

The USART can wake up the system from sleep mode in asynchronous or synchronous mode on any enabled USART interrupt.

In Deep-sleep or power-down mode, you have two options for configuring USART for wake-up:

- If the USART is configured for synchronous slave mode, the USART block can create an interrupt on a received signal even when the USART block receives no clocks from the ARM core - that is in Deep-sleep or Power-down mode.

  As long as the USART receives a clock signal from the master, it can receive up to one byte in the RXDAT register while in Deep-sleep or Power-down mode. Any interrupt raised as part of the receive data process can then wake up the part.
- If the 32 kHz mode is enabled, the USART can run in asynchronous mode using the 32 kHz RTC oscillator and create interrupts.

12.3.2.1 Wake-up from Sleep mode

- Configure the USART in either asynchronous mode or synchronous mode. See Table 180.
- 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 183).

12.3.2.2 Wake-up from Deep-sleep or Power-down mode

- Configure the USART in synchronous slave mode. See Table 180. You must connect the SCLK function to a pin and connect the pin to the master. Alternatively, you can enable the 32 kHz mode and use the USART in asynchronous mode with the 32 kHz RTC oscillator.
- Enable the USART interrupt in the STARTERP1 register. See Table 66 “Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description”.
- Enable the USART interrupt in the NVIC.
- In the PDAWAKE register, configure all peripherals that need to be running when the part wakes up.
- The USART wakes up the part from Deep-sleep or Power-down mode on all events that cause an interrupt and are also enabled in the INTENSET register. Typical wake-up events are:
  - A start bit has been received.
  - The RXDAT buffer has received a byte.
  - Data is ready to be transmitted in the TXDAT buffer and a serial clock from the master has been received.
  - A change in the state of the CTS pin if the CTS function is connected and the DELTACTS interrupt is enabled. This event wakes up the part without the need for either of the two clocks (the 32 kHz clock or the synchronous UART clock) running.

Remark: By enabling or disabling the interrupt in the INTENSET register (Table 183), you can customize when the wake-up occurs in the USART receive/transmit protocol.

12.4 Pin description

Table 178. USART pin description

Function	Direction	Description
U1_TXD	O	Transmitter output for USART1. Serial transmit data.
U1_RXD	I	Receiver input for USART1.
$\overline{\text{U1\_RTS}}$	O	Request To Send output for USART0. This signal supports inter-processor communication through the use of hardware flow control. This feature is active when the USART RTS signal is configured to appear on a device pin.
$\overline{\text{U1\_CTS}}$	I	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).
U1_SCLK	I/O	Serial clock input/output for USART1 in synchronous mode.
U2_TXD	O	Transmitter output for USART2. Serial transmit data.
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.
U3_TXD	O	Transmitter output for USART3. Serial transmit data.
U3_RXD	I	Receiver input for USART3.
$\overline{\text{U3\_RTS}}$	O	Request To Send output for USART3.
$\overline{\text{U3\_CTS}}$	I	Clear To Send input for USART3.
U3_SCLK	I/O	Serial clock input/output for USART3 in synchronous mode.
U4_TXD	O	Transmitter output for USART4. Serial transmit data.
U4_RXD	I	Receiver input for USART4.
$\overline{\text{U4\_RTS}}$	O	Request To Send output for USART4.
$\overline{\text{U4\_CTS}}$	I	Clear To Send input for USART4.
U4_SCLK	I/O	Serial clock input/output for USART4 in synchronous mode.

12.5 General description

The USART receiver block monitors the serial input line, Un_RXD, for valid input. The receiver shift register assembles characters as they are received, after which they are passed to the receiver buffer register to await access by the CPU or the DMA controller.

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

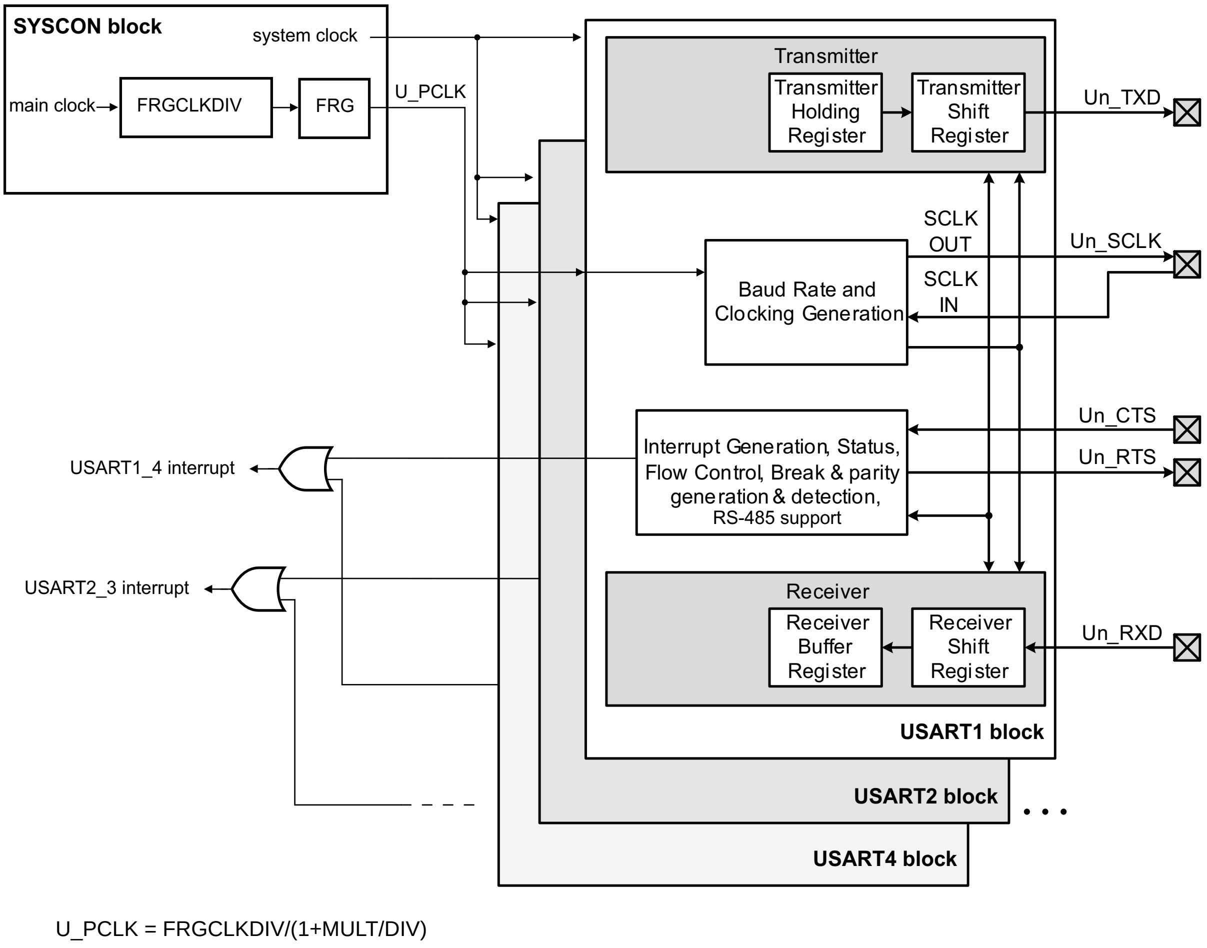
The USART transmitter block accepts data written by the CPU or DMA controllers and buffers the data in the transmit holding register. When the transmitter is available, the transmit shift register takes that data, formats it, and serializes it to the serial output, Un_TXD.

The Baud Rate Generator block divides the incoming clock to create a 16x baud rate clock in the standard asynchronous operating mode. The BRG clock input source is the shared Fractional Rate Generator that runs from the common USART peripheral clock U_PCLK). The 32 kHz operating mode generates a specially timed internal clock based on the RTC oscillator frequency.

In synchronous slave mode, data is transmitted and received using the serial clock directly. In synchronous master mode, data is transmitted and received using the baud rate clock without division.

Status information from the transmitter and receiver is saved and provided via the Stat register. Many of the status flags are able to generate interrupts, as selected by software.

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



U_PCLK = FRGCLKDIV/(1+MULT/DIV)

Fig 26. USART block diagram

12.6 Register description

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

Table 179: Register overview: USART (base address 0x4006 C000 (USART1), 0x4007 0000 (USART2), 0x4007 4000 (USART3), 0x4004 C000 (USART4))

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 180
CTL	R/W	0x004	USART Control register. USART control settings that are more likely to change during operation.	0	Table 181
STAT	R/W	0x008	USART Status register. The complete status value can be read here. Writing ones clears some bits in the register. Some bits can be cleared by writing a 1 to them.	0x000E	Table 182
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 183
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 184
RXDAT	R	0x014	Receiver Data register. Contains the last character received.	-	Table 185
RXDATSTAT	R	0x018	Receiver Data with Status register. Combines the last character received with the current USART receive status. Allows DMA or software to recover incoming data and status together.	-	Table 186
TXDAT	R/W	0x01C	Transmit Data register. Data to be transmitted is written here.	0	Table 187
BRG	R/W	0x020	Baud Rate Generator register. 16-bit integer baud rate divisor value.	0	Table 188
INTSTAT	R	0x024	Interrupt status register. Reflects interrupts that are currently enabled.	0x0005	Table 189
OSR	R/W	0x028	Oversample selection register for asynchronous communication.	0xF	Table 190
ADDR	R/W	0x02C	Address register for automatic address matching.	0	Table 191

12.6.1 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 180. USART Configuration register (CFG, address 0x4006 C000 (USART1), 0x4007 0000 (USART2), 0x4007 4000 (USART3), 0x4004 C000 (USART4)) bit description

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

Table 180. USART Configuration register (CFG, address 0x4006 C000 (USART1), 0x4007 0000 (USART2), 0x4007 4000 (USART3), 0x4004 C000 (USART4)) bit description

Bit	Symbol	Value	Description	Reset Value
7	MODE32K		Selects standard or 32 kHz clocking mode.	0
		0	UART uses standard clocking.	
		1	UART uses the 32 kHz clock from the RTC oscillator as the clock source to the BRG, and uses a special bit clocking scheme.	
8	-		Reserved. Read value is undefined, only zero should be written.	NA
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 12.7.4 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.	
17:16	-		Reserved. Read value is undefined, only zero should be written.	NA

Table 180. USART Configuration register (CFG, address 0x4006 C000 (USART1), 0x4007 0000 (USART2), 0x4007 4000 (USART3), 0x4004 C000 (USART4)) bit description

Bit	Symbol	Value	Description	Reset Value
18	OETA		Output Enable Turnaround time enable for RS-485 operation.	0
		0	Deasserted. If selected by OESEL, the Output Enable signal deasserted at the end of the last stop bit of a transmission.	
		1	Asserted. If selected by OESEL, the Output Enable signal remains asserted for 1 character time after then end the last stop bit of a transmission. OE will also remain asserted if another transmit begins before it is deasserted.	
19	AUTOADDR		Automatic Address matching enable.	0
		0	Disabled. When addressing is enabled by ADDRDET, address matching is done by software. This provides the possibility of versatile addressing (e.g. respond to more than one address).	
		1	Enabled. When addressing is enabled by ADDRDET, address matching is done by hardware, using the value in the ADDR register as the address to match.	
20	OESEL		Output Enable Select.	0
		0	Flow control. The RTS signal is used as the standard flow control function.	
		1	Output enable. The RTS signal is taken over in order to provide an output enable signal to control an RS-485 transceiver.	
21	OEPOL		Output Enable Polarity.	0
		0	Low. If selected by OESEL, the output enable is active low.	
		1	High. If selected by OESEL, the output enable is active high.	
22	RXPOL		Receive data polarity.	0
		0	Not changed. The RX signal is used as it arrives from the pin. This means that the RX rest value is 1, start bit is 0, data is not inverted, and the stop bit is 1.	
		1	Inverted. The RX signal is inverted before being used by the UART. This means that the RX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0.	
23	TXPOL		Transmit data polarity.	0
		0	Not changed. The TX signal is sent out without change. This means that the TX rest value is 1, start bit is 0, data is not inverted, and the stop bit is 1.	
		1	Inverted. The TX signal is inverted by the UART before being sent out. This means that the TX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0.	
31:24	-		Reserved. Read value is undefined, only zero should be written.	NA

12.6.2 USART Control register

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

Table 181. USART Control register (CTL, address 0x4006 C004 (USART1), 0x4007 0004 (USART2), 0x4007 4004 (USART3), 0x4004 C004 (USART4)) bit description

Bit	Symbol	Value	Description	Reset value
0	-		Reserved. Read value is undefined, only zero should be written.	NA
1	TXBRKEN		Break Enable.	0
		0	Normal operation.	
		1	Continuous break is sent immediately when this bit is set, and remains until this bit is cleared. A break may be sent without danger of corrupting any currently transmitting character if the transmitter is first disabled (TXDIS in CTL is set) and then waiting for the transmitter to be disabled (TXDISINT in STAT = 1) before writing 1 to TXBRKEN.	
2	ADDRDET		Enable address detect mode.	0
		0	Disabled. The USART presents all incoming data.	
		1	Enabled. The USART receiver ignores incoming data that does not have the most significant bit of the data (typically the 9th bit) = 1. When the data MSB bit = 1, the receiver treats the incoming data normally, generating a received data interrupt. Software can then check the data to see if this is an address that should be handled. If it is, the ADDRDET bit is cleared by software and further incoming data is handled normally.	
5:3	-		Reserved. Read value is undefined, only zero should be written.	NA
6	TXDIS		Transmit Disable.	0
		0	Not disabled. USART transmitter is not disabled.	
		1	Disabled. USART transmitter is disabled after any character currently being transmitted is complete. This feature can be used to facilitate software flow control.	
7	-		Reserved. Read value is undefined, only zero should be written.	NA
8	CC		Continuous Clock generation. By default, SCLK is only output while data is being transmitted in synchronous mode.	0
		0	Clock on character. In synchronous mode, SCLK cycles only when characters are being sent on Un_TXD or to complete a character that is being received.	
		1	Continuous clock. SCLK runs continuously in synchronous mode, allowing characters to be received on Un_RxD independently from transmission on Un_TXD).	
9	CLRCCONRX		Clear Continuous Clock.	0
		0	No effect on the CC bit.	
		1	Auto-clear. The CC bit is automatically cleared when a complete character has been received. This bit is cleared at the same time.	
15:10	-		Reserved. Read value is undefined, only zero should be written.	NA

Table 181. USART Control register (CTL, address 0x4006 C004 (USART1), 0x4007 0004 (USART2), 0x4007 4004 (USART3), 0x4004 C004 (USART4)) bit description

Bit	Symbol	Value	Description	Reset value
16	AUTOBAUD		Autobaud enable.	0
		0	Disabled. UART is in normal operating mode.	
		1	Enabled. UART is in autobaud mode. This bit should only be set when UART is enabled in the CFG register and the UART receiver is idle. The first start bit of RX is measured and used the update the BRG register to match the received data rate. AUTOBAUD is cleared once this process is complete, or if there is an ABERR. This bit can be cleared by software when set, but only when the UART receiver is idle. Disabling the UART in the CFG register also clears the AUTOBAUD bit.	
31:17	-		Reserved. Read value is undefined, only zero should be written.	NA

12.6.3 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 184).

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 182. USART Status register (STAT, address 0x4006 C008 (USART1), 0x4007 0008 (USART2), 0x4007 4008 (USART3), 0x4004 C008 (USART4)) bit description

Bit	Symbol	Description	Reset value	Access [1]
0	RXRDY	Receiver Ready flag. When 1, indicates that data is available to be read from the receiver buffer. Cleared after a read of the RXDAT or RXDATSTAT registers.	0	RO
1	RXIDLE	Receiver Idle. When 0, indicates that the receiver is currently in the process of receiving data. When 1, indicates that the receiver is not currently in the process of receiving data.	1	RO
2	TXRDY	Transmitter Ready flag. When 1, this bit indicates that data may be written to the transmit buffer. Previous data may still be in the process of being transmitted. Cleared when data is written to TXDAT. Set when the data is moved from the transmit buffer to the transmit shift register.	1	RO
3	TXIDLE	Transmitter Idle. When 0, indicates that the transmitter is currently in the process of sending data.When 1, indicate that the transmitter is not currently in the process of sending data.	1	RO
4	CTS	This bit reflects the current state of the CTS signal, regardless of the setting of the CTSEN bit in the CFG register. This will be the value of the CTS input pin unless loopback mode is enabled.	NA	RO
5	DELTACTS	This bit is set when a change in the state is detected for the CTS flag above. This bit is cleared by software.	0	W1

Table 182. USART Status register (STAT, address 0x4006 C008 (USART1), 0x4007 0008 (USART2), 0x4007 4008 (USART3), 0x4004 C008 (USART4)) bit description

Bit	Symbol	Description	Reset value	Access [1]
6	TXDISSTAT	Transmitter Disabled Interrupt flag. When 1, this bit indicates that the USART transmitter is fully idle after being disabled via the TXDIS in the CTL register (TXDIS = 1).	0	RO
7	-	Reserved. Read value is undefined, only zero should be written.	NA	NA
8	OVERRUNINT	Overrun Error interrupt flag. This flag is set when a new character is received while the receiver buffer is still in use. If this occurs, the newly received character in the shift register is lost.	0	W1
9	-	Reserved. Read value is undefined, only zero should be written.	NA	NA
10	RXBRK	Received Break. This bit reflects the current state of the receiver break detection logic. It is set when the Un_RXD pin remains low for 16 bit times. Note that FRAMERRINT will also be set when this condition occurs because the stop bit(s) for the character would be missing. RXBRK is cleared when the Un_RXD pin goes high.	0	RO
11	DELTARXBRK	This bit is set when a change in the state of receiver break detection occurs. Cleared by software.	0	W1
12	START	This bit is set when a start is detected on the receiver input. Its purpose is primarily to allow wake-up from Deep-sleep or Power-down mode immediately when a start is detected. Cleared by software.	0	W1
13	FRAMERRINT	Framing Error interrupt flag. This flag is set when a character is received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.	0	W1
14	PARITYERRINT	Parity Error interrupt flag. This flag is set when a parity error is detected in a received character..	0	W1
15	RXNOISEINT	Received Noise interrupt flag. Three samples of received data are taken in order to determine the value of each received data bit, except in synchronous mode. This acts as a noise filter if one sample disagrees. This flag is set when a received data bit contains one disagreeing sample. This could indicate line noise, a baud rate or character format mismatch, or loss of synchronization during data reception.	0	W1
16	ABERR	Autobaud Error. An autobaud error can occur if the BRG counts to its limit before the end of the start bit that is being measured, essentially an autobaud time-out.	0	W1
31:17	-	Reserved. Read value is undefined, only zero should be written.	NA	NA

[1] RO = Read-only, W1 = write 1 to clear.

12.6.4 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 183. USART Interrupt Enable read and set register (INTENSET, address 0x4006 C00C (USART1), 0x4007 000C (USART2), 0x4007 400C (USART3), 0x4004 C00C (USART4)) bit description

Bit	Symbol	Description	Reset Value
0	RXRDYEN	When 1, enables an interrupt when there is a received character available to be read from the RXDAT register.	0
1	-	Reserved. Read value is undefined, only zero should be written.	NA
2	TXRDYEN	When 1, enables an interrupt when the TXDAT register is available to take another character to transmit.	0
3	TXIDLEEN	When 1, enables an interrupt when the transmitter becomes idle (TXIDLE = 1).	0
4	-	Reserved. Read value is undefined, only zero should be written.	NA
5	DELTACTSEN	When 1, enables an interrupt when there is a change in the state of the CTS input.	0
6	TXDISEN	When 1, enables an interrupt when the transmitter is fully disabled as indicated by the TXDISINT flag in STAT. See description of the TXDISINT bit for details.	0
7	-	Reserved. Read value is undefined, only zero should be written.	NA
8	OVERRUNEN	When 1, enables an interrupt when an overrun error occurred.	0
10:9	-	Reserved. Read value is undefined, only zero should be written.	NA
11	DELTARXBRKEN	When 1, enables an interrupt when a change of state has occurred in the detection of a received break condition (break condition asserted or deasserted).	0
12	STARTEN	When 1, enables an interrupt when a received start bit has been detected.	0
13	FRAMERREN	When 1, enables an interrupt when a framing error has been detected.	0
14	PARITYERREN	When 1, enables an interrupt when a parity error has been detected.	0
15	RXNOISEEN	When 1, enables an interrupt when noise is detected.	0
16	ABERREN	When 1, enables an interrupt when an autobaud error occurs.	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	NA

12.6.5 USART Interrupt Enable Clear register

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

Table 184. USART Interrupt Enable clear register (INTENCLR, address 0x4006 C010 (USART1), 0x4007 0010 (USART2), 0x4007 4010 (USART3), 0x4004 C010 (USART4)) bit description

Bit	Symbol	Description	Reset Value
0	RXRDYCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
1	-	Reserved. Read value is undefined, only zero should be written.	NA
2	TXRDYCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
3	TXIDLECLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
4	-	Reserved. Read value is undefined, only zero should be written.	NA
5	DELTACTSCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
6	TXDISINTCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
7	-	Reserved. Read value is undefined, only zero should be written.	NA
8	OVERRUNCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
10:9	-	Reserved. Read value is undefined, only zero should be written.	NA
11	DELTARXBRKCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
12	STARTCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
13	FRAMERRCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
14	PARITYERRCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
15	RXNOISECLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
16	ABERRCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	NA

12.6.6 USART Receiver Data register

The RXDAT register contains the last character received before any overrun.

Remark: Reading this register changes the status flags in the RXDATSTAT register.

Table 185. USART Receiver Data register (RXDAT, address 0x4006 C014 (USART1), 0x4007 0014 (USART2), 0x4007 4014 (USART3), 0x4004 C014 (USART4)) 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

12.6.7 USART Receiver Data with Status register

The RXDATSTAT register contains the next complete character to be read and its relevant status flags. This allows getting all information related to a received character with one 16-bit read, which may be especially useful when the DMA is used with the USART receiver.

Remark: Reading this register changes the status flags.

Table 186. USART Receiver Data with Status register (RXDATSTAT, address 0x4006 C018 (USART1), 0x4007 0018 (USART2), 0x4007 4018 (USART3), 0x4004 C018 (USART4)) bit description

Bit	Symbol	Description	Reset Value
8:0	RXDAT	The USART Receiver Data register contains the next received character. The number of bits that are relevant depends on the USART configuration settings.	0
12:9	-	Reserved, the value read from a reserved bit is not defined.	NA
13	FRAMERR	Framing Error status flag. This bit is valid when there is a character to be read in the RXDAT register and reflects the status of that character. This bit will set when the character in RXDAT was received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.	0
14	PARITYERR	Parity Error status flag. This bit is valid when there is a character to be read in the RXDAT register and reflects the status of that character. This bit will be set when a parity error is detected in a received character.	0
15	RXNOISE	Received Noise flag. See description of the RXNOISEINT bit in Table 182.	0
31:16	-	Reserved, the value read from a reserved bit is not defined.	NA

12.6.8 USART Transmitter Data Register

The TXDAT register is written in order to send data via the USART transmitter. That data will be transferred to the transmit shift register when it is available, and another character may then be written to TXDAT.

Table 187. USART Transmitter Data Register (TXDAT, address 0x4006 C01C (USART1), 0x4007 001C (USART2), 0x4007 401C (USART3), 0x4004 C01C (USART4)) 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

12.6.9 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.

Note that in 32 kHz mode, the baud rate generator is still used and must be set to 0 if 9600 baud is required.

Details on how to select the right values for BRG can be found in Section 12.7.1.

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 188. USART Baud Rate Generator register (BRG, address 0x4006 C020 (USART1), 0x4007 0020 (USART2), 0x4007 4020 (USART3), 0x4004 C020 (USART4)) 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

12.6.10 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 182 for detailed descriptions of the interrupt flags.

Table 189. USART Interrupt Status register (INTSTAT, address 0x4006 C024 (USART1), 0x4007 0024 (USART2), 0x4007 4024 (USART3), 0x4004 C024 (USART4)) bit description

Bit	Symbol	Description	Reset Value
0	RXRDY	Receiver Ready flag.	0
1	-	Reserved. Read value is undefined, only zero should be written.	NA
2	TXRDY	Transmitter Ready flag.	1
3	TXIDLE	Transmitter idle status.	1
4	-	Reserved. Read value is undefined, only zero should be written.	NA
5	DELTACTS	This bit is set when a change in the state of the CTS input is detected.	0
6	TXDISINT	Transmitter Disabled Interrupt flag.	0
7	-	Reserved. Read value is undefined, only zero should be written.	NA
8	OVERRUNINT	Overrun Error interrupt flag.	0
10:9	-	Reserved. Read value is undefined, only zero should be written.	NA
11	DELTARXBRK	This bit is set when a change in the state of receiver break detection occurs.	0
12	START	This bit is set when a start is detected on the receiver input.	0
13	FRAMERRINT	Framing Error interrupt flag.	0
14	PARITYERRINT	Parity Error interrupt flag.	0
15	RXNOISEINT	Received Noise interrupt flag.	0
16	ABERR	Autobaud Error flag.	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	NA

12.6.11 Oversample selection register

The OSR register allows selection of oversampling in asynchronous modes. The oversample value is the number of BRG clocks used to receive one data bit. The default is industry standard 16x oversampling.

Changing the oversampling can sometimes allow better matching of baud rates in cases where the peripheral clock rate is not a multiple of 16 times the expected maximum baud rate. For all modes where the OSR setting is used, the UART receiver takes three consecutive samples of input data in the approximate middle of the bit time. Smaller values of OSR can make the sampling position within a data bit less accurate and may potentially cause more noise errors or incorrect data.

Table 190. Oversample selection register (OSR, address 0x4006 C028 (USART1), 0x4007 0028 (USART2), 0x4007 4028 (USART3), 0x4004 C028 (USART4)) bit description

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

12.6.12 Address register

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

Table 191. Address register (ADDR, address 0x4006 C02C (USART1), 0x4007 002C (USART2), 0x4007 402C (USART3), 0x4004 C02C (USART4)) bit description

Bit	Symbol	Description	Reset value
7:0	ADDRESS	8-bit address used with automatic address matching. Used when address detection is enabled (ADDRDET in CTL = 1) and automatic address matching is enabled (AUTOADDR in CFG = 1).	0
31:8	-	Reserved, the value read from a reserved bit is not defined.	NA

12.7 Functional description

12.7.1 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 25.

12.7.1.1 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 44 “UART Fractional baud rate clock divider register (FRGCLKDIV, address 0x4004 80A0) bit description”, Table 51 “USART fractional generator divider value register (UARTFRGDIV, address 0x4004 80F0) bit description”, and Table 52 “USART fractional generator multiplier value register (UARTFRGMULT, address 0x4004 80F4) bit description”.

For details see Section 12.3.1 “Configure the USART clock and baud rate”.

12.7.1.2 Baud Rate Generator (BRG)

The Baud Rate Generator (see Section 12.6.9) 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.

12.7.1.3 Baud rate calculations

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

12.7.1.4 32 kHz mode

In order to use a 32 kHz clock(32.768 kHz) to operate a USART at any reasonable speed, a number of adaptations need to be made. First, 16x overclocking has to be abandoned. Otherwise, the maximum data rate would be very low. For the same reason, multiple samples of each data bit must be reduced to one. Finally, special clocking has to be used for individual bit times because 32 kHz is not particularly close to an even multiple of any standard baud rate.

When 32 kHz mode is enabled, clocking comes from the RTC oscillator. The FRG is bypassed, and the BRG can be used to divide down the default 9600 baud to lower rates. Other adaptations required to make the UART work for rates up to 9600 baud are done internally. Rate error will be less than one half percent in this mode, provided the RTC oscillator is operating at the intended frequency of 32.768 kHz.

12.7.2 DMA

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

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

12.7.3 Synchronous mode

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

12.7.4 Flow control

The USART supports both hardware and software flow control.

12.7.4.1 Hardware flow control

The USART supports hardware flow control using RTS and/or CTS signaling. 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 27 shows an overview of RTS and CTS within the USART.

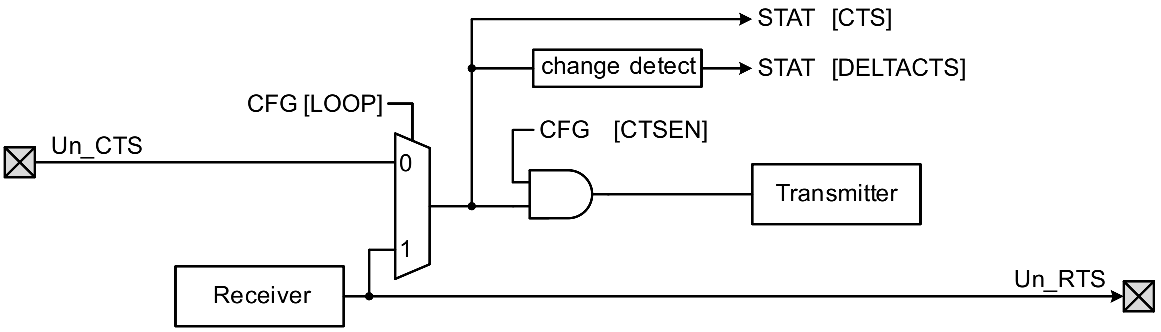


Fig 27. Hardware flow control using RTS and CTS

12.7.4.2 Software flow control

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

12.7.5 Autobaud function

The autobaud functions attempts to measure the start bit time of the next received character. For this to work, the measured character must have a 1 in the least significant bit position, so that the start bit is bounded by a falling and rising edge. The measurement is made using the current clocking settings, including the oversampling configuration. The result is that a value is stored in the BRG register that is as close as possible to the correct setting for the sampled character and the current clocking settings. The sampled character is provided in the RXDAT and RXDATSTAT registers, allowing software to double check for the expected character.

Autobaud includes a time-out that is flagged by ABERR if no character is received at the expected time. It is recommended that autobaud only be enabled when the USART receiver is idle. Once enabled, either RXRDY or ABERR will be asserted at some point. The assertion of RXRDY clears the AUTOBAUD bit automatically. The assertion of ABERR clears the AUTOBAUD bit once the receive line goes inactive.

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

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

12.7.6 RS-485 support

This USART has provisions for hardware address recognition (see the AUTOADDR bit in the CFG register in Section 12.6.1 and the ADDR register in Section 12.6.12), as well as software address recognition (see the ADDRDET bit in the CTL register in Section 12.6.2).

Automatic data direction control with the RTS pin can be set up using the OESEL, OEPOL and OETA bits in the CFG register (Section 12.6.1). Data direction control can also be implemented in software using a GPIO pin.

12.7.7 Oversampling

Typical industry standard UARTs use a 16x oversample clock to transmit and receive asynchronous data. This is the number of BRG clocks used for one data bit. The Oversample Select Register (OSR) allows this UART to use a 16x down to a 5x oversample clock. There is no oversampling in synchronous modes.

Reducing the oversampling can sometimes help in getting better baud rate matching when the baud rate is very high, or the peripheral clock is very low. For example, the closest actual rate near 115,200 baud with a 12 MHz peripheral clock and 16x oversampling is 107,143 baud, giving a rate error of 7%. Changing the oversampling to 15x gets the actual rate to 114,286 baud, a rate error of 0.8%. Reducing the oversampling to 13x gets the actual rate to 115,385 baud, a rate error of only 0.16%.

There is a cost for altering the oversampling. In asynchronous modes, the UART takes three samples of incoming data on consecutive oversample clocks, as close to the center of a bit time as can be done. When the oversample rate is reduced, the three samples spread out and occupy a larger proportion of a bit time. For example, with 5x oversampling, there is one oversample clock, then three data samples taken, then one more oversample clock before the end of the bit time. Since the oversample clock is running asynchronously from the input data, skew of the input data relative to the expected timing has little room for error. At 16x oversampling, there are several oversample clocks before actual data sampling is done, making the sampling more robust. Generally speaking, it is recommended to use the highest oversampling where the rate error is acceptable in the system.

UM10732

Chapter 13: LPC11U6x/E6x I2C0/1-bus interfaces

Rev. 1.8 — 19 December 2016 **User manual**

13.1 How to read this chapter

I2C0 and I2C1 are available on all parts.

13.2 Features

- Standard I^{2}C-compliant bus interfaces may be configured as Master, Slave, or Master/Slave.
- Arbitration is handled between simultaneously transmitting masters without corruption of serial data on the bus.
- Programmable clock allows adjustment of I^{2}C transfer rates.
- Data transfer is bidirectional between masters and slaves.
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus.
- Serial clock synchronization is used as a handshake mechanism to suspend and resume serial transfer.
- Supports Fast-mode Plus.
- Optional recognition of up to four distinct slave addresses.
- Monitor mode allows observing all I^{2}C-bus traffic, regardless of slave address.
- I^{2}C-bus can be used for test and diagnostic purposes.
- The I^{2}C0-bus contains a standard I^{2}C-compliant bus interface with two open-drain pins.

13.3 Basic configuration

The I^{2}C-bus interface is configured using the following registers:

1. Pins: The I2C pin functions and the I2C mode are configured in the IOCON register block (Table 90).
2. Power and peripheral clock: In the SYSAHBCLKCTRL register, set bit 5 (Table 40).
3. Reset: Before accessing the I2C block, ensure that the I2C0_RST_N and I2C1_RST_N bits in the PRESETCTRL register (Table 23) are set to 1. This de-asserts the reset signal to the I2C0 and I2C1 blocks.

13.4 General description

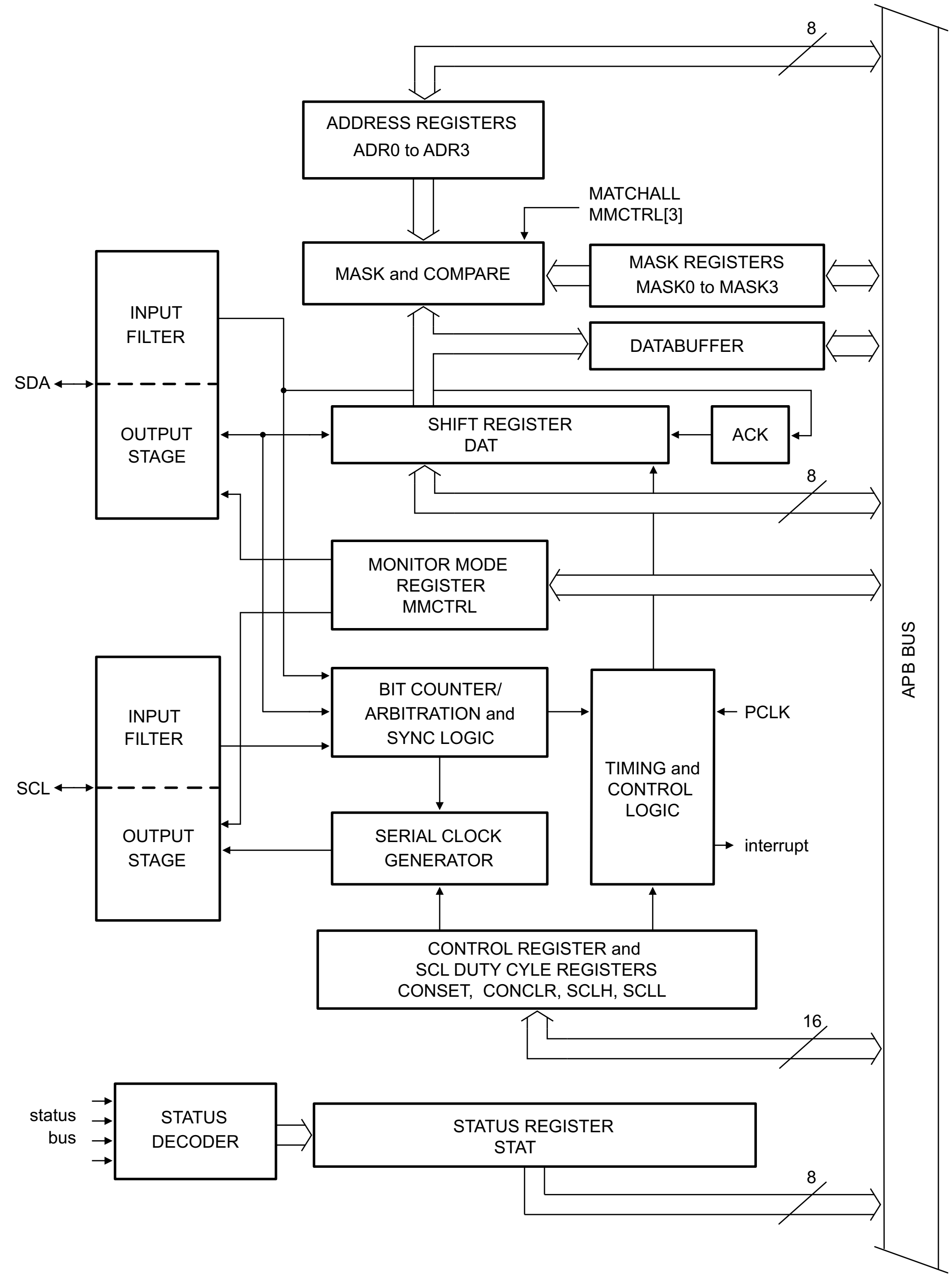


Fig 28. I^{2}C serial interface block diagram

13.4.1 Address Registers, ADR0 to ADR3

These registers may be loaded with the 7-bit slave address (7 most significant bits) to which the I^{2}C block will respond when programmed as a slave transmitter or receiver. The LSB (GC) is used to enable General Call address (0x00) recognition. When multiple slave addresses are enabled, the actual address received may be read from the DAT register at the state where the own slave address has been received.

13.4.2 Address mask registers, MASK0 to MASK3

The four mask registers each contain seven active bits (7:1). Any bit in these registers which is set to ‘1’ will cause an automatic compare on the corresponding bit of the received address when it is compared to the ADRn register associated with that mask register. In other words, bits in an ADRn register which are masked are not taken into account in determining an address match.

When an address-match interrupt occurs, the processor will have to read the data register (DAT) to determine what the received address was that actually caused the match.

13.4.3 Comparator

The comparator compares the received 7-bit slave address with its own slave address (7 most significant bits in ADR). It also compares the first received 8-bit byte with the General Call address (0x00). If an equality is found, the appropriate status bits are set and an interrupt is requested.

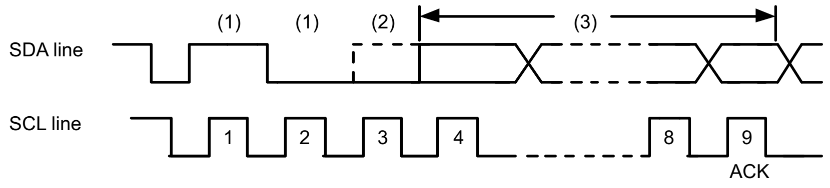
13.4.4 Shift register, DAT

This 8-bit register contains a byte of serial data to be transmitted or a byte which has just been received. Data in DAT is always shifted from right to left; the first bit to be transmitted is the MSB (bit 7) and, after a byte has been received, the first bit of received data is located at the MSB of DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; DAT always contains the last byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in DAT.

13.4.5 Arbitration and synchronization logic

In the master transmitter mode, the arbitration logic checks that every transmitted logic 1 actually appears as a logic 1 on the I^{2}C-bus. If another device on the bus overrules a logic 1 and pulls the SDA line low, arbitration is lost, and the I^{2}C block immediately changes from master transmitter to slave receiver. The I^{2}C block will continue to output clock pulses (on SCL) until transmission of the current serial byte is complete.

Arbitration may also be lost in the master receiver mode. Loss of arbitration in this mode can only occur while the I^{2}C block is returning a “not acknowledge: (logic 1) to the bus. Arbitration is lost when another device on the bus pulls this signal low. Since this can occur only at the end of a serial byte, the I^{2}C block generates no further clock pulses. Figure 29 shows the arbitration procedure.



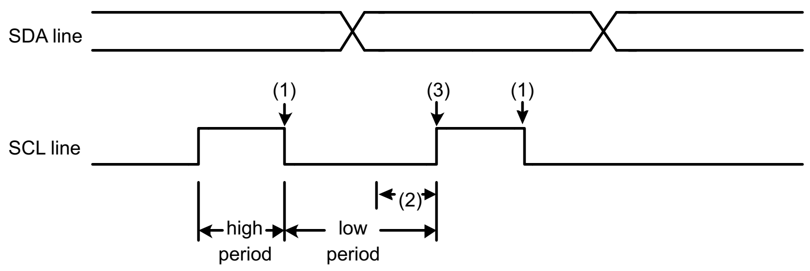
(1) Another device transmits serial data.

(2) Another device overrules a logic (dotted line) transmitted this I^2C master by pulling the SDA line low. Arbitration is lost, and this I^2C enters Slave Receiver mode.

(3) This I^2C is in Slave Receiver mode but still generates clock pulses until the current byte has been transmitted. This I^2C will not generate clock pulses for the next byte. Data on SDA originates from the new master once it has won arbitration.

Fig 29. Arbitration procedure

The synchronization logic will synchronize the serial clock generator with the clock pulses on the SCL line from another device. If two or more master devices generate clock pulses, the “mark” duration is determined by the device that generates the shortest “marks,” and the “space” duration is determined by the device that generates the longest “spaces”. Figure 30 shows the synchronization procedure.



(1) Another device pulls the SCL line low before this I^2C has timed a complete high time. The other device effectively determines the (shorter) HIGH period.

(2) Another device continues to pull the SCL line low after this I^2C has timed a complete low time and released SCL. The I^2C clock generator is forced to wait until SCL goes HIGH. The other device effectively determines the (longer) LOW period.

(3) The SCL line is released , and the clock generator begins timing the HIGH time.

Fig 30. Serial clock synchronization

A slave may stretch the space duration to slow down the bus master. The space duration may also be stretched for handshaking purposes. This can be done after each bit or after a complete byte transfer. the I^2C block will stretch the SCL space duration after a byte has been transmitted or received and the acknowledge bit has been transferred. The serial interrupt flag (SI) is set, and the stretching continues until the serial interrupt flag is cleared.

13.4.6 Serial clock generator

This programmable clock pulse generator provides the SCL clock pulses when the I^2C block is in the master transmitter or master receiver mode. It is switched off when the I^2C block is in slave mode. The I^2C output clock frequency and duty cycle is programmable via the I^{2}C Clock Control Registers. See the description of the I2CSCLL and I2CSCLH registers for details. The output clock pulses have a duty cycle as programmed unless the bus is synchronizing with other SCL clock sources as described above.

13.4.7 Timing and control

The timing and control logic generates the timing and control signals for serial byte handling. This logic block provides the shift pulses for DAT, enables the comparator, generates and detects START and STOP conditions, receives and transmits acknowledge bits, controls the master and slave modes, contains interrupt request logic, and monitors the I^{2}C-bus status.

13.4.8 Control register, CONSET and CONCLR

The I^{2}C control register contains bits used to control the following I^{2}C block functions: start and restart of a serial transfer, termination of a serial transfer, bit rate, address recognition, and acknowledgment.

The contents of the I^{2}C control register may be read as CONSET. Writing to CONSET will set bits in the I^{2}C control register that correspond to ones in the value written. Conversely, writing to CONCLR will clear bits in the I^{2}C control register that correspond to ones in the value written.

13.4.9 Status decoder and status register

The status decoder takes all of the internal status bits and compresses them into a 5-bit code. This code is unique for each I^{2}C-bus status. The 5-bit code may be used to generate vector addresses for fast processing of the various service routines. Each service routine processes a particular bus status. There are 26 possible bus states if all four modes of the I^{2}C block are used. The 5-bit status code is latched into the five most significant bits of the status register when the serial interrupt flag is set (by hardware) and remains stable until the interrupt flag is cleared by software. The three least significant bits of the status register are always zero. If the status code is used as a vector to service routines, then the routines are displaced by eight address locations. Eight bytes of code is sufficient for most of the service routines (see the software example in this section).

13.4.10 I^{2}C operating modes

In a given application, the I^{2}C block may operate as a master, a slave, or both. In the slave mode, the I^{2}C hardware looks for any one of its four slave addresses and the General Call address. If one of these addresses is detected, an interrupt is requested. If the processor wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave operation is not interrupted. If bus arbitration is lost in the master mode, the I^{2}C block switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

13.4.10.1 Master Transmitter mode

In this mode data is transmitted from master to slave. Before the master transmitter mode can be entered, the CONSET register must be initialized as shown in Table 192. I2EN must be set to 1 to enable the I^{2}C function. If the AA bit is 0, the I^{2}C interface will not acknowledge any address when another device is master of the bus, so it can not enter slave mode. The STA, STO and SI bits must be 0. The SI Bit is cleared by writing 1 to the SIC bit in the CONCLR register. THe STA bit should be cleared after writing the slave address.

Table 192. CONSET used to configure Master mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	0	-	-

The first byte transmitted contains the slave address of the receiving device (7 bits) and the data direction bit. In this mode the data direction bit (R/W) should be 0 which means Write. The first byte transmitted contains the slave address and Write bit. Data is transmitted 8 bits at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

The I^{2}C interface will enter master transmitter mode when software sets the STA bit. The I^{2}C logic will send the START condition as soon as the bus is free. After the START condition is transmitted, the SI bit is set, and the status code in the STAT register is 0x08. This status code is used to vector to a state service routine which will load the slave address and Write bit to the DAT register, and then clear the SI bit. SI is cleared by writing a 1 to the SIC bit in the CONCLR register.

When the slave address and R/W bit have been transmitted and an acknowledgment bit has been received, the SI bit is set again, and the possible status codes now are 0x18, 0x20, or 0x38 for the master mode, or 0x68, 0x78, or 0xB0 if the slave mode was enabled (by setting AA to 1). The appropriate actions to be taken for each of these status codes are shown in Table 210 to Table 215.

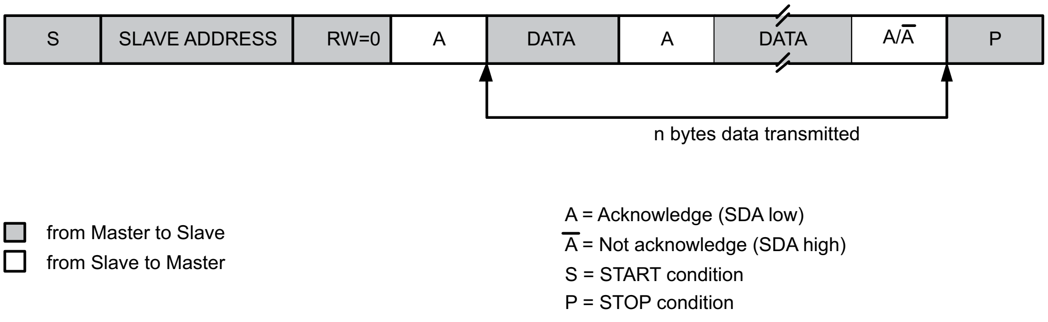


Fig 31. Format in the Master Transmitter mode

13.4.10.2 Master Receiver mode

In the master receiver mode, data is received from a slave transmitter. The transfer is initiated in the same way as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load the slave address and the data direction bit to the I^{2}C Data register (DAT), and then clear the SI bit. In this case, the data direction bit (R/W) should be 1 to indicate a read.

When the slave address and data direction bit have been transmitted and an acknowledge bit has been received, the SI bit is set, and the Status Register will show the status code. For master mode, the possible status codes are 0x40, 0x48, or 0x38. For slave mode, the possible status codes are 0x68, 0x78, or 0xB0. For details, refer to Table 211.

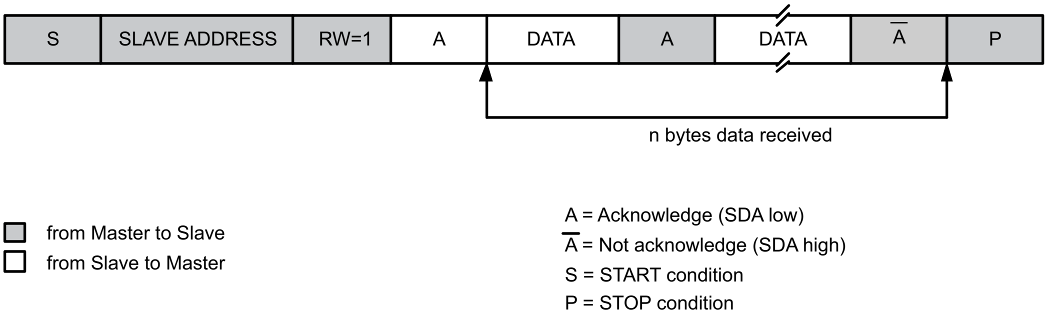


Fig 32. Format of Master Receiver mode

After a Repeated START condition, I²C may switch to the master transmitter mode.

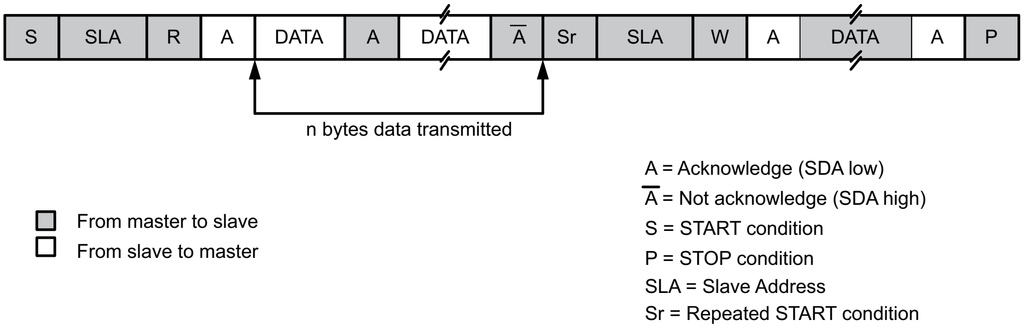


Fig 33. A Master Receiver switches to Master Transmitter after sending Repeated START

13.4.10.3 Slave Receiver mode

In the slave receiver mode, data bytes are received from a master transmitter. To initialize the slave receiver mode, write any of the Slave Address registers (ADR0-3) and write the I²C Control Set register (CONSET) as shown in Table 193.

Table 193. CONSET used to configure Slave mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	1	-	-

I2EN must be set to 1 to enable the I²C function. AA bit must be set to 1 to acknowledge its own slave address or the General Call address. The STA, STO and SI bits are set to 0.

After ADR and CONSET are initialized, the I²C interface waits until it is addressed by its own address or general address followed by the data direction bit. If the direction bit is 0 (W), it enters slave receiver mode. If the direction bit is 1 (R), it enters slave transmitter mode. After the address and direction bit have been received, the SI bit is set and a valid status code can be read from the Status register (STAT). Refer to Table 214 for the status codes and actions.

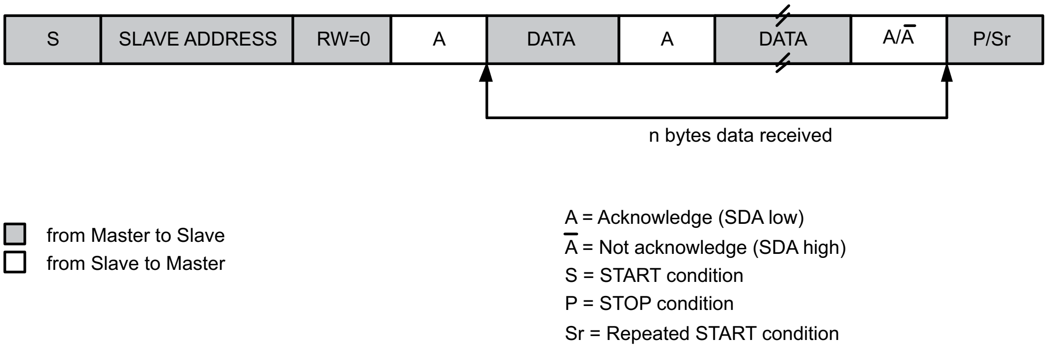


Fig 34. Format of Slave Receiver mode

13.4.10.4 Slave Transmitter mode

The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will be 1, indicating a read operation. Serial data is transmitted via SDA while the serial clock is input through SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer. In a given application, I^{2}C may operate as a master and as a slave. In the slave mode, the I^{2}C hardware looks for its own slave address and the General Call address. If one of these addresses is detected, an interrupt is requested. When the microcontrollers wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave action is not interrupted. If bus arbitration is lost in the master mode, the I^{2}C interface switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

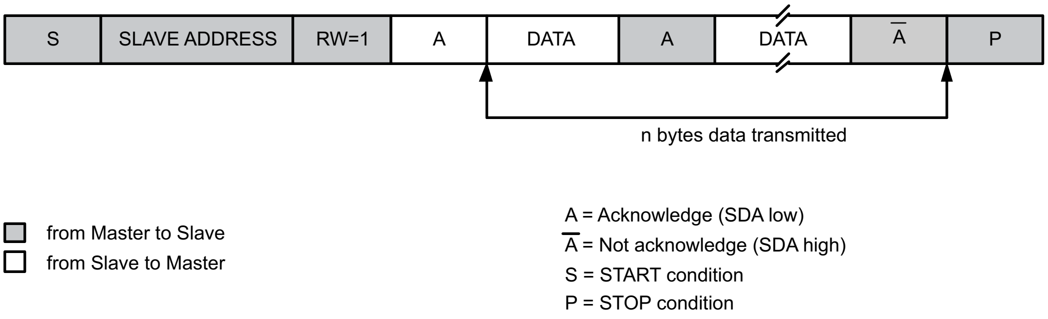


Fig 35. Format of Slave Transmitter mode

13.4.11 I2C bus configuration

A typical I^{2}C-bus configuration is shown in Figure 36. Depending on the state of the direction bit (R/W), two types of data transfers are possible on the I^{2}C-bus:

- Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte.

- Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a “not acknowledge” is returned. The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a Repeated START condition. Since a Repeated START condition is also the beginning of the next serial transfer, the I^{2}C bus will not be released.

The I^{2}C interface is byte oriented and has four operating modes: master transmitter mode, master receiver mode, slave transmitter mode and slave receiver mode.

The I^{2}C interface complies with the entire I^{2}C specification, supporting the ability to turn power off to the ARM core without interfering with other devices on the same I^{2}C-bus.

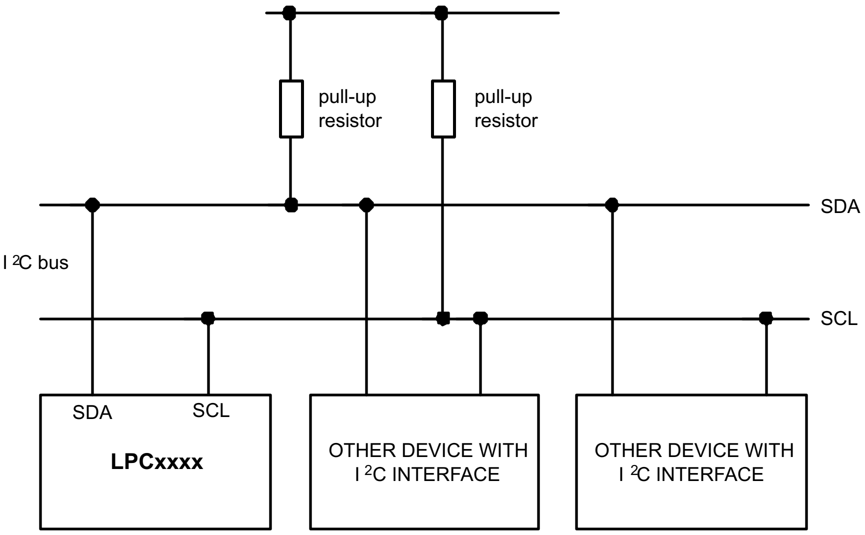


Fig 36. I^{2}C-bus configuration

13.4.12 I^{2}C Fast-mode Plus

Fast-Mode Plus supports a 1 Mbit/sec transfer rate.

13.4.13 Applications

Interfaces to external I^{2}C standard parts, such as serial RAMs, LCDs, tone generators, other microcontrollers, etc.

13.4.14 Input filters and output stages

Input signals are synchronized with the internal clock, and spikes shorter than three clocks are filtered out.

The output for I^{2}C is a special pad designed to conform to the I^{2}C specification.

13.5 Pin description

Table 194. I^{2}C-bus pin description

Pin	Type	Description
I2C0_SDA	Input/Output	I^{2}C0 Serial Data. This is an open-drain pin. Fast-mode Plus, fast, and standard data rates supported.
I2C0_SCL	Input/Output	I^{2}C Serial Clock. This is an open-drain pin. Fast-mode Plus, fast, and standard data rates supported.
I2C1_SDA	Input/Output	I^{2}C0 Serial Data. This is a standard digital pin. Only fast and standard data rates supported.
I2C1_SCL	Input/Output	I^{2}C Serial Clock. This is a standard digital pin. Only fast and standard data rates supported.

The I^{2}C0-bus pins must be configured through the IOCON_PIO0_4 and IOCON_PIO0_5 (Table 90) registers for Standard/ Fast-mode or Fast-mode Plus. In Fast-mode Plus, rates above 400 kHz and up to 1 MHz may be selected. The I^{2}C0-bus pins are open-drain outputs and fully compatible with the I^{2}C-bus specification.

13.6 Register description

Table 195. Register overview: I^{2}C (base address 0x4000 0000 (I2C0), 0x4002 0000 (I2C1))

Name	Access	Address offset	Description	Reset value[1]	Reference
CONSET	R/W	0x000	**I2C Control Set Register.** When a one is written to a bit of this register, the corresponding bit in the I^{2}C control register is set. Writing a zero has no effect on the corresponding bit in the I^{2}C control register.	0x00	Table 196
STAT	RO	0x004	**I2C Status Register.** During I^{2}C operation, this register provides detailed status codes that allow software to determine the next action needed.	0xF8	Table 197
DAT	R/W	0x008	**I2C Data Register.** During master or slave transmit mode, data to be transmitted is written to this register. During master or slave receive mode, data that has been received may be read from this register.	0x00	Table 198
ADR0	R/W	0x00C	**I2C Slave Address Register 0.** Contains the 7-bit slave address for operation of the I^{2}C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	0x00	Table 199
SCLH	R/W	0x010	**SCH Duty Cycle Register High Half Word.** Determines the high time of the I^{2}C clock.	0x04	Table 200
SCLL	R/W	0x014	**SCL Duty Cycle Register Low Half Word.** Determines the low time of the I^{2}C clock. I2nSCLL and I2nSCLH together determine the clock frequency generated by an I^{2}C master and certain times used in slave mode.	0x04	Table 201
CONCLR	WO	0x018	**I2C Control Clear Register.** When a one is written to a bit of this register, the corresponding bit in the I^{2}C control register is cleared. Writing a zero has no effect on the corresponding bit in the I^{2}C control register.	NA	Table 203
MMCTRL	R/W	0x01C	**Monitor mode control register.**	0x00	Table 204

Table 195. Register overview: I^{2}C (base address 0x4000 0000 (I2C0), 0x4002 0000 (I2C1)) *…continued*

Name	Access	Address offset	Description	Reset value[1]	Reference
ADR1	R/W	0x020	**I2C Slave Address Register 1.** Contains the 7-bit slave address for operation of the I^{2}C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	0x00	Table 205
ADR2	R/W	0x024	**I2C Slave Address Register 2.** Contains the 7-bit slave address for operation of the I^{2}C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	0x00	Table 205
ADR3	R/W	0x028	**I2C Slave Address Register 3.** Contains the 7-bit slave address for operation of the I^{2}C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	0x00	Table 205
DATA_BUFFER	RO	0x02C	**Data buffer register.** The contents of the 8 MSBs of the I2DAT shift register will be transferred to the DATA_BUFFER automatically after every nine bits (8 bits of data plus ACK or NACK) has been received on the bus.	0x00	Table 206
MASK0	R/W	0x030	**I2C Slave address mask register 0**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	0x00	Table 207
MASK1	R/W	0x034	**I2C Slave address mask register 1**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	0x00	Table 207
MASK2	R/W	0x038	**I2C Slave address mask register 2**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	0x00	Table 207
MASK3	R/W	0x03C	**I2C Slave address mask register 3**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	0x00	Table 207

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

13.6.1 I^{2}C Control Set register

The CONSET registers control setting of bits in the CON register that controls operation of the I^{2}C interface. Writing a one to a bit of this register causes the corresponding bit in the I^{2}C control register to be set. Writing a zero has no effect.

Table 196. I^{2}C Control Set register (CONSET, address 0x4000 0000 (I2C0) and 0x4002 0000 (I2C1)) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AA	Assert acknowledge flag.	
3	SI	I^{2}C interrupt flag.	0
4	STO	STOP flag.	0
5	STA	START flag.	0
6	I2EN	I^{2}C interface enable.	0
31:7	-	Reserved. The value read from a reserved bit is not defined.	-

I2EN I^{2}C Interface Enable. When I2EN is 1, the I^{2}C interface is enabled. I2EN can be cleared by writing 1 to the I2ENC bit in the CONCLR register. When I2EN is 0, the I^{2}C interface is disabled.

When I2EN is “0”, the SDA and SCL input signals are ignored, the I^{2}C block is in the “not addressed” slave state, and the STO bit is forced to “0”.

I2EN should not be used to temporarily release the I^{2}C-bus since, when I2EN is reset, the I^{2}C-bus status is lost. The AA flag should be used instead.

STA is the START flag. Setting this bit causes the I^{2}C interface to enter master mode and transmit a START condition or transmit a Repeated START condition if it is already in master mode.

When STA is 1 and the I^{2}C interface is not already in master mode, it enters master mode, checks the bus and generates a START condition if the bus is free. If the bus is not free, it waits for a STOP condition (which will free the bus) and generates a START condition after a delay of a half clock period of the internal clock generator. If the I^{2}C interface is already in master mode and data has been transmitted or received, it transmits a Repeated START condition. STA may be set at any time, including when the I^{2}C interface is in an addressed slave mode.

STA can be cleared by writing 1 to the STAC bit in the CONCLR register. When STA is 0, no START condition or Repeated START condition will be generated.

If STA and STO are both set, then a STOP condition is transmitted on the I^{2}C-bus if it the interface is in master mode, and transmits a START condition thereafter. If the I^{2}C interface is in slave mode, an internal STOP condition is generated, but is not transmitted on the bus.

STO is the STOP flag. Setting this bit causes the I^{2}C interface to transmit a STOP condition in master mode, or recover from an error condition in slave mode. When STO is 1 in master mode, a STOP condition is transmitted on the I^{2}C-bus. When the bus detects the STOP condition, STO is cleared automatically.

In slave mode, setting this bit can recover from an error condition. In this case, no STOP condition is transmitted to the bus. The hardware behaves as if a STOP condition has been received and it switches to “not addressed” slave receiver mode. The STO flag is cleared by hardware automatically.

SI is the I^{2}C Interrupt Flag. This bit is set when the I^{2}C state changes. However, entering state F8 does not set SI since there is nothing for an interrupt service routine to do in that case.

While SI is set, the low period of the serial clock on the SCL line is stretched, and the serial transfer is suspended. When SCL is HIGH, it is unaffected by the state of the SI flag. SI must be reset by software, by writing a 1 to the SIC bit in the CONCLR register.

AA is the Assert Acknowledge Flag. When set to 1, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. The address in the Slave Address Register has been received.
2. The General Call address has been received while the General Call bit (GC) in the ADR register is set.
3. A data byte has been received while the I^{2}C is in the master receiver mode.
4. A data byte has been received while the I^{2}C is in the addressed slave receiver mode

The AA bit can be cleared by writing 1 to the AAC bit in the CONCLR register. When AA is 0, a not acknowledge (HIGH level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. A data byte has been received while the I^{2}C is in the master receiver mode.
2. A data byte has been received while the I^{2}C is in the addressed slave receiver mode.

13.6.2 I^{2}C Status register

Each I^{2}C Status register reflects the condition of the corresponding I^{2}C interface. The I^{2}C Status register is Read-Only.

Table 197. I^{2}C Status register (STAT, address 0x4000 0004 (I2C0) and 0x4002 0004 (I2C1)) bit description

Bit	Symbol	Description	Reset value
2:0	-	These bits are unused and are always 0.	0
7:3	Status	These bits give the actual status information about the I^{2}C interface.	0x1F
31:8	-	Reserved. The value read from a reserved bit is not defined.	-

The three least significant bits are always 0. Taken as a byte, the status register contents represent a status code. There are 26 possible status codes. When the status code is 0xF8, there is no relevant information available and the SI bit is not set. All other 25 status codes correspond to defined I^{2}C states. When any of these states entered, the SI bit will be set. For a complete list of status codes, refer to tables from Table 210 to Table 215.

13.6.3 I^{2}C Data register

This register contains the data to be transmitted or the data just received. The CPU can read and write to this register only while it is not in the process of shifting a byte, when the SI bit is set. Data in DAT register remains stable as long as the SI bit is set. Data in DAT register is always shifted from right to left: the first bit to be transmitted is the MSB (bit 7), and after a byte has been received, the first bit of received data is located at the MSB of the DAT register.

Table 198. I^2C Data register (DAT, address 0x4000 0008 (I2C0) and 0x4002 0008 (I2C1)) bit description

Bit	Symbol	Description	Reset value
7:0	Data	This register holds data values that have been received or are to be transmitted.	0
31:8	-	Reserved. The value read from a reserved bit is not defined.	-

13.6.4 I^2C Slave Address register 0

This register is readable and writable and are only used when an I^2C interface is set to slave mode. In master mode, this register has no effect. The LSB of the ADR register is the General Call bit. When this bit is set, the General Call address (0x00) is recognized.

If this register contains 0x00, the I^2C will not acknowledge any address on the bus. All four registers (ADR0 to ADR3) will be cleared to this disabled state on reset. See also Table 205.

Table 199. I^2C Slave Address register 0 (ADR0, address 0x4000 000C (I2C0) and 0x4002 000C (I2C1)) bit description

Bit	Symbol	Description	Reset value
0	GC	General Call enable bit.	0
7:1	Address	The I^2C device address for slave mode.	0x00
31:8	-	Reserved. The value read from a reserved bit is not defined.	-

13.6.5 I^2C SCL HIGH and LOW duty cycle registers

Table 200. I^2C SCL HIGH Duty Cycle register (SCLH, address 0x4000 0010 (I2C0) and 0x4002 0010 (I2C1)) bit description

Bit	Symbol	Description	Reset value
15:0	SCLH	Count for SCL HIGH time period selection.	0x0004
31:16	-	Reserved. The value read from a reserved bit is not defined.	-

Table 201. I^2C SCL Low duty cycle register (SCLL, address 0x4000 0014 (I2C0) and 0x4002 0014 (I2C1)) bit description

Bit	Symbol	Description	Reset value
15:0	SCLL	Count for SCL low time period selection.	0x0004
31:16	-	Reserved. The value read from a reserved bit is not defined.	-

13.6.5.1 Selecting the appropriate I^2C data rate and duty cycle

Software must set values for the registers SCLH and SCLL to select the appropriate data rate and duty cycle. SCLH defines the number of I2C_PCLK cycles for the SCL HIGH time, SCLL defines the number of I2C_PCLK cycles for the SCL low time. The frequency is determined by the following formula (I2C_PCLK is the frequency of the peripheral I2C clock):

$$I^2C_{bitfrequency} = \frac{I2CPCLK}{SCLH + SCLL} \tag{6}$$

The values for SCLL and SCLH must ensure that the data rate is in the appropriate I^{2}C data rate range. Each register value must be greater than or equal to 4. Table 202 gives some examples of I^{2}C-bus rates based on I2C_PCLK frequency and SCLL and SCLH values.

Table 202. SCLL + SCLH values for selected I^{2}C clock values

I^{2}C mode	I^{2}C bit frequency	I2C_PCLK (MHz)								
		6	8	10	12	16	20	30	40	50
		SCLH + SCLL								
Standard mode	100 kHz	60	80	100	120	160	200	300	400	500
Fast-mode	400 kHz	15	20	25	30	40	50	75	100	125
Fast-mode Plus	1 MHz	-	8	10	12	16	20	30	40	50

SCLL and SCLH values should not necessarily be the same. Software can set different duty cycles on SCL by setting these two registers. For example, the I^{2}C-bus specification defines the SCL low time and high time at different values for a Fast-mode and Fast-mode Plus I^{2}C.

13.6.6 I^{2}C Control Clear register

The CONCLR register control clearing of bits in the CON register that controls operation of the I^{2}C interface. Writing a one to a bit of this register causes the corresponding bit in the I^{2}C control register to be cleared. Writing a zero has no effect.

Table 203. I^{2}C Control Clear register (CONCLR, address 0x4000 0018 (I2C0) and 0x4002 0018 (I2C1)) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AAC	Assert acknowledge Clear bit.	
3	SIC	I^{2}C interrupt Clear bit.	0
4	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
5	STAC	START flag Clear bit.	0
6	I2ENC	I^{2}C interface Disable bit.	0
7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
31:8	-	Reserved. The value read from a reserved bit is not defined.	-

AAC is the Assert Acknowledge Clear bit. Writing a 1 to this bit clears the AA bit in the CONSET register. Writing 0 has no effect.

SIC is the I^{2}C Interrupt Clear bit. Writing a 1 to this bit clears the SI bit in the CONSET register. Writing 0 has no effect.

STAC is the START flag Clear bit. Writing a 1 to this bit clears the STA bit in the CONSET register. Writing 0 has no effect.

I**2ENC** is the I^{2}C Interface Disable bit. Writing a 1 to this bit clears the I2EN bit in the CONSET register. Writing 0 has no effect.

13.6.7 I^{2}C Monitor mode control register

This register controls the Monitor mode which allows the I^{2}C module to monitor traffic on the I^{2}C bus without actually participating in traffic or interfering with the I^{2}C bus.

Table 204. I^{2}C Monitor mode control register (MMCTRL, address 0x4000 001C (I2C0) and 0x4002 001C (I2C1)) bit description

Bit	Symbol	Value	Description	Reset value
0	MM_ENA		Monitor mode enable.	0
		0	Disable. Monitor mode disabled.	
		1	Enabled. The I^{2}C module will enter monitor mode. In this mode the SDA output will be forced high. This will prevent the I^{2}C module from outputting data of any kind (including ACK) onto the I^{2}C data bus. Depending on the state of the ENA_SCL bit, the output may be also forced high, preventing the module from having control over the I^{2}C clock line.	
1	ENA_SCL		SCL output enable.	0
		0	High. When this bit is cleared to 0, the SCL output will be forced high when the module is in monitor mode. As described above, this will prevent the module from having any control over the I^{2}C clock line.	
		1	Stretch. When this bit is set, the I^{2}C module may exercise the same control over the clock line that it would in normal operation. This means that, acting as a slave peripheral, the I^{2}C module can stretch the clock line (hold it low) until it has had time to respond to an I^{2}C interrupt. When the ENA_SCL bit is cleared and the I^{2}C no longer has the ability to stall the bus, interrupt response time becomes important. To give the part more time to respond to an I^{2}C interrupt under these conditions, a DATA _BUFFER register is used to hold received data for a full 9-bit word transmission time.	

Table 204. I^{2}C Monitor mode control register (MMCTRL, address 0x4000 001C (I2C0) and 0x4002 001C (I2C1)) bit description

Bit	Symbol	Value	Description	Reset value
2	MATCH_ALL		Select interrupt register match.	0
		0	Match address. When this bit is cleared, an interrupt will only be generated when a match occurs to one of the (up-to) four address registers described above. That is, the module will respond as a normal slave as far as address-recognition is concerned.	
		1	Any address. When this bit is set to 1 and the I^{2}C is in monitor mode, an interrupt will be generated on ANY address received. This will enable the part to monitor all traffic on the bus.	
31:3	-	-	Reserved. The value read from reserved bits is not defined.	

Remark: The ENA_SCL and MATCH_ALL bits have no effect if the MM_ENA is 0 (i.e. if the module is NOT in monitor mode).

13.6.7.1 Interrupt in Monitor mode

All interrupts will occur as normal when the module is in monitor mode. This means that the first interrupt will occur when an address-match is detected (any address received if the MATCH_ALL bit is set, otherwise an address matching one of the four address registers).

Subsequent to an address-match detection, interrupts will be generated after each data byte is received for a slave-write transfer, or after each byte that the module “thinks” it has transmitted for a slave-read transfer. In this second case, the data register will actually contain data transmitted by some other slave on the bus which was actually addressed by the master.

Following all of these interrupts, the processor may read the data register to see what was actually transmitted on the bus.

13.6.7.2 Loss of arbitration in Monitor mode

In monitor mode, the I^{2}C module will not be able to respond to a request for information by the bus master or issue an ACK). Some other slave on the bus will respond instead. This will most probably result in a lost-arbitration state as far as our module is concerned.

Software should be aware of the fact that the module is in monitor mode and should not respond to any loss of arbitration state that is detected. In addition, hardware may be designed into the module to block some/all loss of arbitration states from occurring if those state would either prevent a desired interrupt from occurring or cause an unwanted interrupt to occur. Whether any such hardware will be added is still to be determined.

13.6.8 I^{2}C Slave Address registers

These registers are readable and writable and are only used when an I^{2}C interface is set to slave mode. In master mode, this register has no effect. The LSB of the ADR register is the General Call bit. When this bit is set, the General Call address (0x00) is recognized.

If these registers contain 0x00, the I^{2}C will not acknowledge any address on the bus. All four registers will be cleared to this disabled state on reset (also see Table 199).

Table 205. I^{2}C Slave Address registers (ADR[1:3], address 0x4000 0020 (ADR1) to 0x4000 0028 (ADR3) (I2C0) and 0x4002 0020 (ADR1) to 0x4002 0028 (ADR3) (I2C1)) bit description

Bit	Symbol	Description	Reset value
0	GC	General Call enable bit.	0
7:1	Address	The I^{2}C device address for slave mode.	0x00
31:8	-	Reserved. The value read from a reserved bit is not defined.	0

13.6.9 I^{2}C Data buffer register

In monitor mode, the I^{2}C module may lose the ability to stretch the clock (stall the bus) if the ENA_SCL bit is not set. This means that the processor will have a limited amount of time to read the contents of the data received on the bus. If the processor reads the DAT shift register, as it ordinarily would, it could have only one bit-time to respond to the interrupt before the received data is overwritten by new data.

To give the processor more time to respond, a new 8-bit, read-only DATA_BUFFER register will be added. The contents of the 8 MSBs of the DAT shift register will be transferred to the DATA_BUFFER automatically after every nine bits (8 bits of data plus ACK or NACK) has been received on the bus. This means that the processor will have nine bit transmission times to respond to the interrupt and read the data before it is overwritten.

The processor will still have the ability to read the DAT register directly, as usual, and the behavior of DAT will not be altered in any way.

Although the DATA_BUFFER register is primarily intended for use in monitor mode with the ENA_SCL bit = ‘0’, it will be available for reading at any time under any mode of operation.

Table 206. I^{2}C Data buffer register (DATA_BUFFER, address 0x4000 002C (I2C0) and 0x4002 002C (I2C1)) bit description

Bit	Symbol	Description	Reset value
7:0	Data	This register holds contents of the 8 MSBs of the DAT shift register.	0
31:8	-	Reserved. The value read from a reserved bit is not defined.	0

13.6.10 I^{2}C Mask registers

The four mask registers each contain seven active bits (7:1). Any bit in these registers which is set to ‘1’ will cause an automatic compare on the corresponding bit of the received address when it is compared to the ADRn register associated with that mask register. In other words, bits in an ADRn register which are masked are not taken into account in determining an address match.

On reset, all mask register bits are cleared to ‘0’.

The mask register has no effect on comparison to the General Call address (“0000000”).

Bits(31:8) and bit(0) of the mask registers are unused and should not be written to. These bits will always read back as zeros.

When an address-match interrupt occurs, the processor will have to read the data register (DAT) to determine what the received address was that actually caused the match.

Table 207. I^{2}C Mask registers (MASK[0:3], 0x4000 0030 (MASK0) to 0x4000 003C (MASK3) (I2C0) and 0x4002 0030 (MASK0) to 0x4002 003C (MASK3) (I2C1)) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved. User software should not write ones to reserved bits. This bit reads always back as 0.	0
7:1	MASK	Mask bits.	0x00
31:8	-	Reserved. The value read from reserved bits is undefined.	0

13.7 Functional description

13.7.1 Details of I^{2}C operating modes

The four operating modes are:

- Master Transmitter
- Master Receiver
- Slave Receiver
- Slave Transmitter

Data transfers in each mode of operation are shown in Figure 37, Figure 38, Figure 39, Figure 40, and Figure 41. Table 208 lists abbreviations used in these figures when describing the I^{2}C operating modes.

Table 208. Abbreviations used to describe an I^{2}C operation

Abbreviation	Explanation
S	START Condition
SLA	7-bit slave address
R	Read bit (HIGH level at SDA)
W	Write bit (LOW level at SDA)
A	Acknowledge bit (LOW level at SDA)
$\overline{A}$	Not acknowledge bit (HIGH level at SDA)
Data	8-bit data byte
P	STOP condition

In Figure 37 to Figure 41, circles are used to indicate when the serial interrupt flag is set. The numbers in the circles show the status code held in the STAT register. At these points, a service routine must be executed to continue or complete the serial transfer. These service routines are not critical since the serial transfer is suspended until the serial interrupt flag is cleared by software.

When a serial interrupt routine is entered, the status code in STAT is used to branch to the appropriate service routine. For each status code, the required software action and details of the following serial transfer are given in tables from Table 210 to Table 216.

13.7.1.1 Master Transmitter mode

In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see Figure 37). Before the master transmitter mode can be entered, I2CON must be initialized as follows:

Table 209. CONSET used to initialize Master Transmitter mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	x	-	-

The I^{2}C rate must also be configured in the SCLL and SCLH registers. I2EN must be set to logic 1 to enable the I^{2}C block. If the AA bit is reset, the I^{2}C block will not acknowledge its own slave address or the General Call address in the event of another device becoming master of the bus. In other words, if AA is reset, the I^{2}C interface cannot enter slave mode. STA, STO, and SI must be reset.

The master transmitter mode may now be entered by setting the STA bit. The I^{2}C logic will now test the I^{2}C-bus and generate a START condition as soon as the bus becomes free. When a START condition is transmitted, the serial interrupt flag (SI) is set, and the status code in the status register (STAT) will be 0x08. This status code is used by the interrupt service routine to enter the appropriate state service routine that loads DAT with the slave address and the data direction bit (SLA+W). The SI bit in CON must then be reset before the serial transfer can continue.

When the slave address and the direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in STAT are possible. There are 0x18, 0x20, or 0x38 for the master mode and also 0x68, 0x78, or 0xB0 if the slave mode was enabled (AA = logic 1). The appropriate action to be taken for each of these status codes is detailed in Table 210. After a Repeated START condition (state 0x10). The I^{2}C block may switch to the master receiver mode by loading DAT with SLA+R).

Table 210. Master Transmitter mode

Status Code (I2CSTAT)	Status of the I²C-bus and hardware	Application software response					Next action taken by I²C hardware
		To/From DAT	To CON				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+W; clear STA	X	0	0	X	SLA+W will be transmitted; ACK bit will be received.
0x10	A Repeated START condition has been transmitted.	Load SLA+W or	X	0	0	X	As above.
		Load SLA+R; Clear STA	X	0	0	X	SLA+R will be transmitted; the I²C block will be switched to MST/REC mode.
0x18	SLA+W has been transmitted; ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No DAT action or	1	0	0	X	Repeated START will be transmitted.
		No DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x20	SLA+W has been transmitted; NOT ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No DAT action or	1	0	0	X	Repeated START will be transmitted.
		No DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x28	Data byte in DAT has been transmitted; ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No DAT action or	1	0	0	X	Repeated START will be transmitted.
		No DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x30	Data byte in DAT has been transmitted; NOT ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No DAT action or	1	0	0	X	Repeated START will be transmitted.
		No DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x38	Arbitration lost in SLA+R/W or Data bytes.	No DAT action or	0	0	0	X	I²C-bus will be released; not addressed slave will be entered.
		No DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.

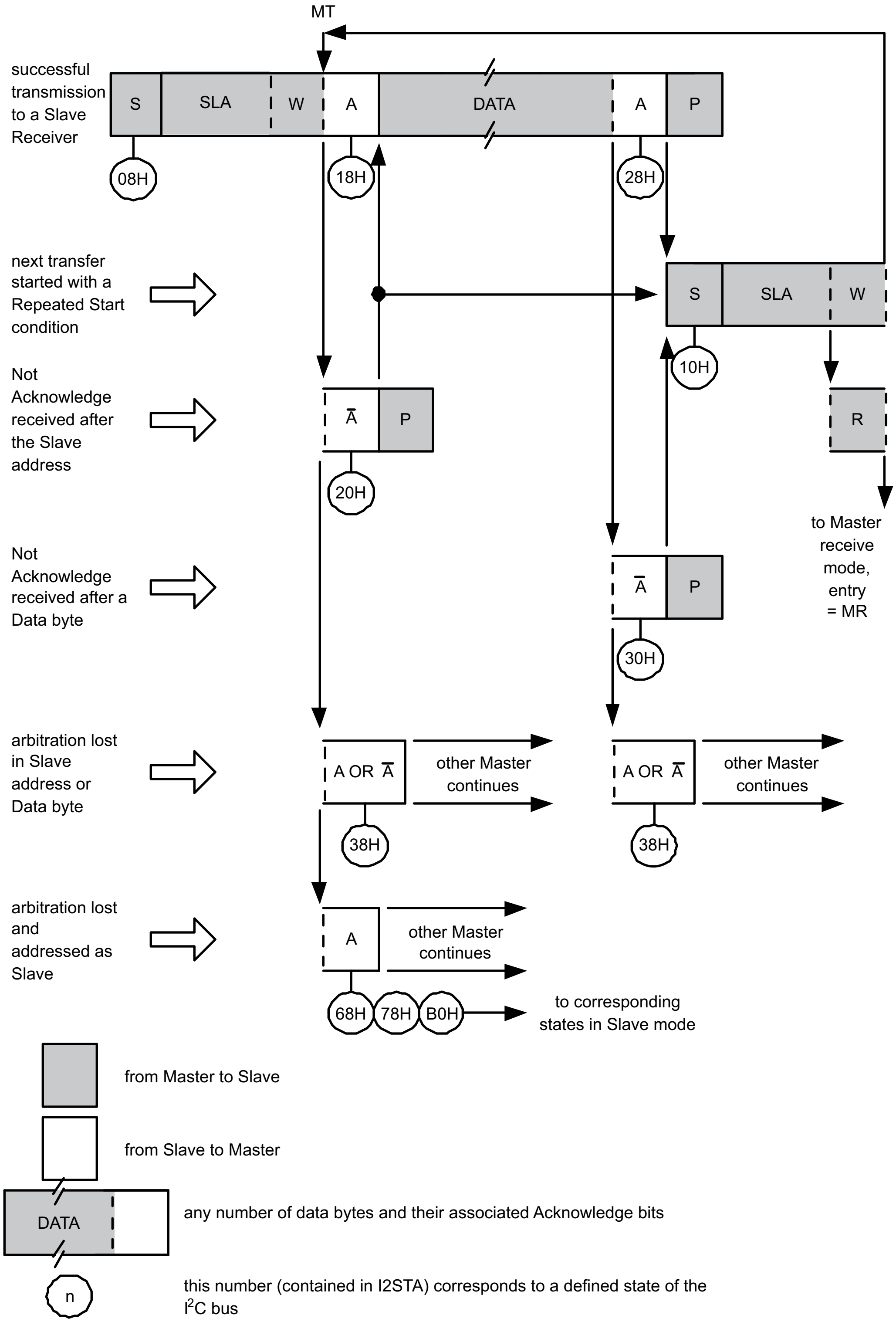


Fig 37. Format and states in the Master Transmitter mode

13.7.1.2 Master Receiver mode

In the master receiver mode, a number of data bytes are received from a slave transmitter (see Figure 38). The transfer is initialized as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load DAT with the 7-bit slave address and the data direction bit (SLA+R). The SI bit in CON must then be cleared before the serial transfer can continue.

When the slave address and the data direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in STAT are possible. These are 0x40, 0x48, or 0x38 for the master mode and also 0x68, 0x78, or 0xB0 if the slave mode was enabled (AA = 1). The appropriate action to be taken for each of these status codes is detailed in Table 211. After a Repeated START condition (state 0x10), the I^{2}C block may switch to the master transmitter mode by loading DAT with SLA+W.

Table 211. Master Receiver mode

Status Code (STAT)	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From DAT	To CON				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+R	X	0	0	X	SLA+R will be transmitted; ACK bit will be received.
0x10	A Repeated START condition has been transmitted.	Load SLA+R or	X	0	0	X	As above.
		Load SLA+W	X	0	0	X	SLA+W will be transmitted; the I^2C block will be switched to MST/TRX mode.
0x38	Arbitration lost in NOT ACK bit.	No DAT action or	0	0	0	X	I^2C-bus will be released; the I^2C block will enter slave mode.
		No DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.
0x40	SLA+R has been transmitted; ACK has been received.	No DAT action or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		No DAT action	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x48	SLA+R has been transmitted; NOT ACK has been received.	No DAT action or	1	0	0	X	Repeated START condition will be transmitted.
		No DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x50	Data byte has been received; ACK has been returned.	Read data byte or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		Read data byte	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x58	Data byte has been received; NOT ACK has been returned.	Read data byte or	1	0	0	X	Repeated START condition will be transmitted.
		Read data byte or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		Read data byte	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.

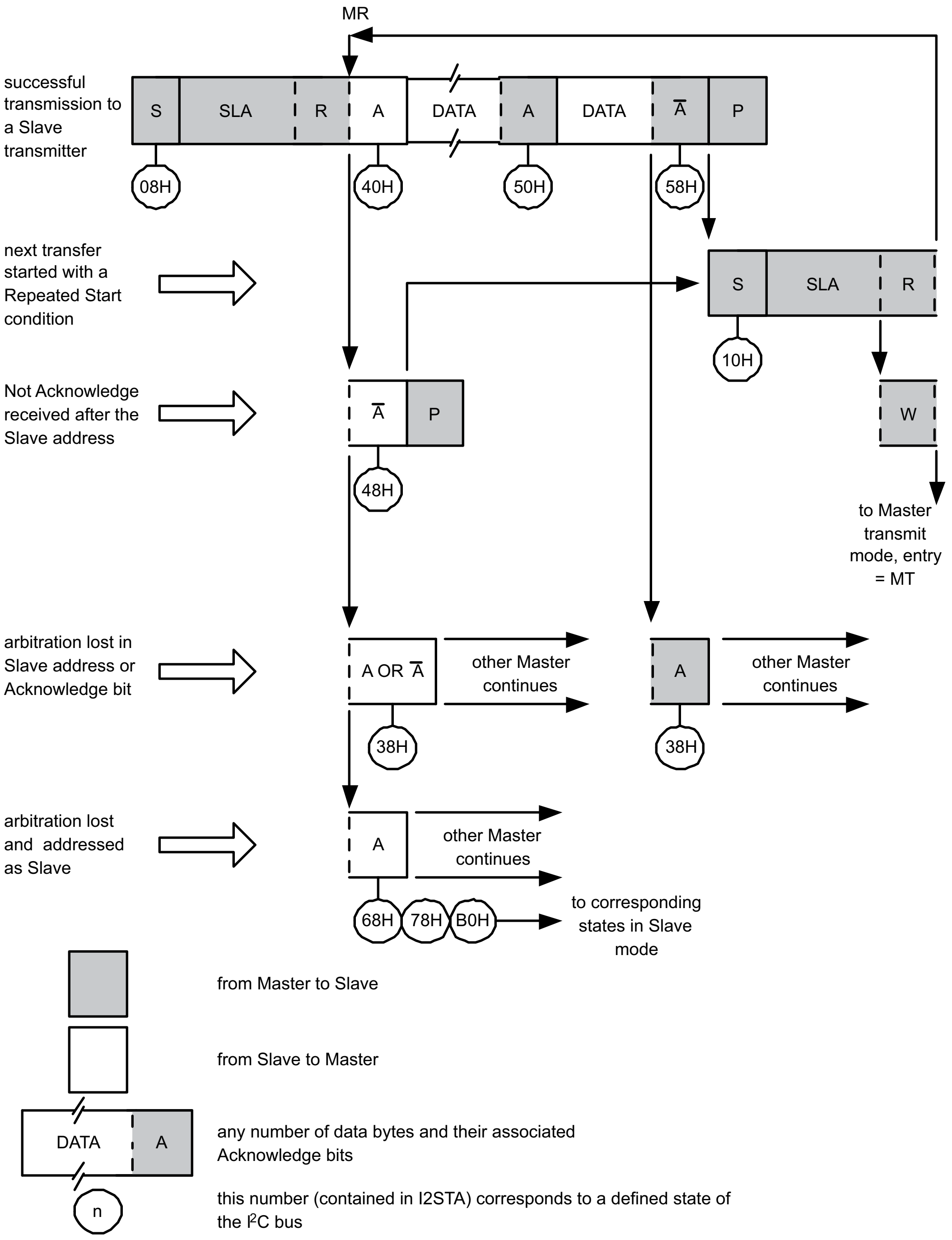


Fig 38. Format and states in the Master Receiver mode

13.7.1.3 Slave Receiver mode

In the slave receiver mode, a number of data bytes are received from a master transmitter (see Figure 39). To initiate the slave receiver mode, ADR and CON must be loaded as follows:

Table 212. ADR usage in Slave Receiver mode

Bit	7	6	5	4	3	2	1	0
Symbol	own slave 7-bit address							GC

The upper 7 bits are the address to which the I^{2}C block will respond when addressed by a master. If the LSB (GC) is set, the I^{2}C block will respond to the General Call address (0x00); otherwise it ignores the General Call address.

Table 213. CONSET used to initialize Slave Receiver mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	1	-	-

The I^{2}C-bus rate settings do not affect the I^{2}C block in the slave mode. I2EN must be set to logic 1 to enable the I^{2}C block. The AA bit must be set to enable the I^{2}C block to acknowledge its own slave address or the General Call address. STA, STO, and SI must be reset.

When ADR and CON have been initialized, the I^{2}C block waits until it is addressed by its own slave address followed by the data direction bit which must be “0” (W) for the I^{2}C block to operate in the slave receiver mode. After its own slave address and the W bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from STAT. This status code is used to vector to a state service routine. The appropriate action to be taken for each of these status codes is detailed in Table 214. The slave receiver mode may also be entered if arbitration is lost while the I^{2}C block is in the master mode (see status 0x68 and 0x78).

If the AA bit is reset during a transfer, the I^{2}C block will return a not acknowledge (logic 1) to SDA after the next received data byte. While AA is reset, the I^{2}C block does not respond to its own slave address or a General Call address. However, the I^{2}C-bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I^{2}C block from the I^{2}C-bus.

Table 214. Slave Receiver mode

Status Code (STAT)	Status of the I²C-bus and hardware	Application software response					Next action taken by I²C hardware
		To/From DAT	To CON				
			STA	STO	SI	AA	
0x60	Own SLA+W has been received; ACK has been returned.	No DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x68	Arbitration lost in SLA+R/W as master; Own SLA+W has been received, ACK returned.	No DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x70	General call address (0x00) has been received; ACK has been returned.	No DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x78	Arbitration lost in SLA+R/W as master; General call address has been received, ACK has been returned.	No DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x80	Previously addressed with own SLV address; DATA has been received; ACK has been returned.	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.
0x88	Previously addressed with own SLA; DATA byte has been received; NOT ACK has been returned.	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		Read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if ADR[0] = logic 1.
		Read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0x90	Previously addressed with General Call; DATA byte has been received; ACK has been returned.	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.

Table 214. Slave Receiver mode *…continued*

Status Code (STAT)	Status of the I²C-bus and hardware	Application software response					Next action taken by I²C hardware
		To/From DAT	To CON				
			STA	STO	SI	AA	
0x98	Previously addressed with General Call; DATA byte has been received; NOT ACK has been returned.	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		Read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if ADR[0] = logic 1.
		Read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0xA0	A STOP condition or Repeated START condition has been received while still addressed as SLV/REC or SLV/TRX.	No STDAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		No STDAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if ADR[0] = logic 1.
		No STDAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		No STDAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.

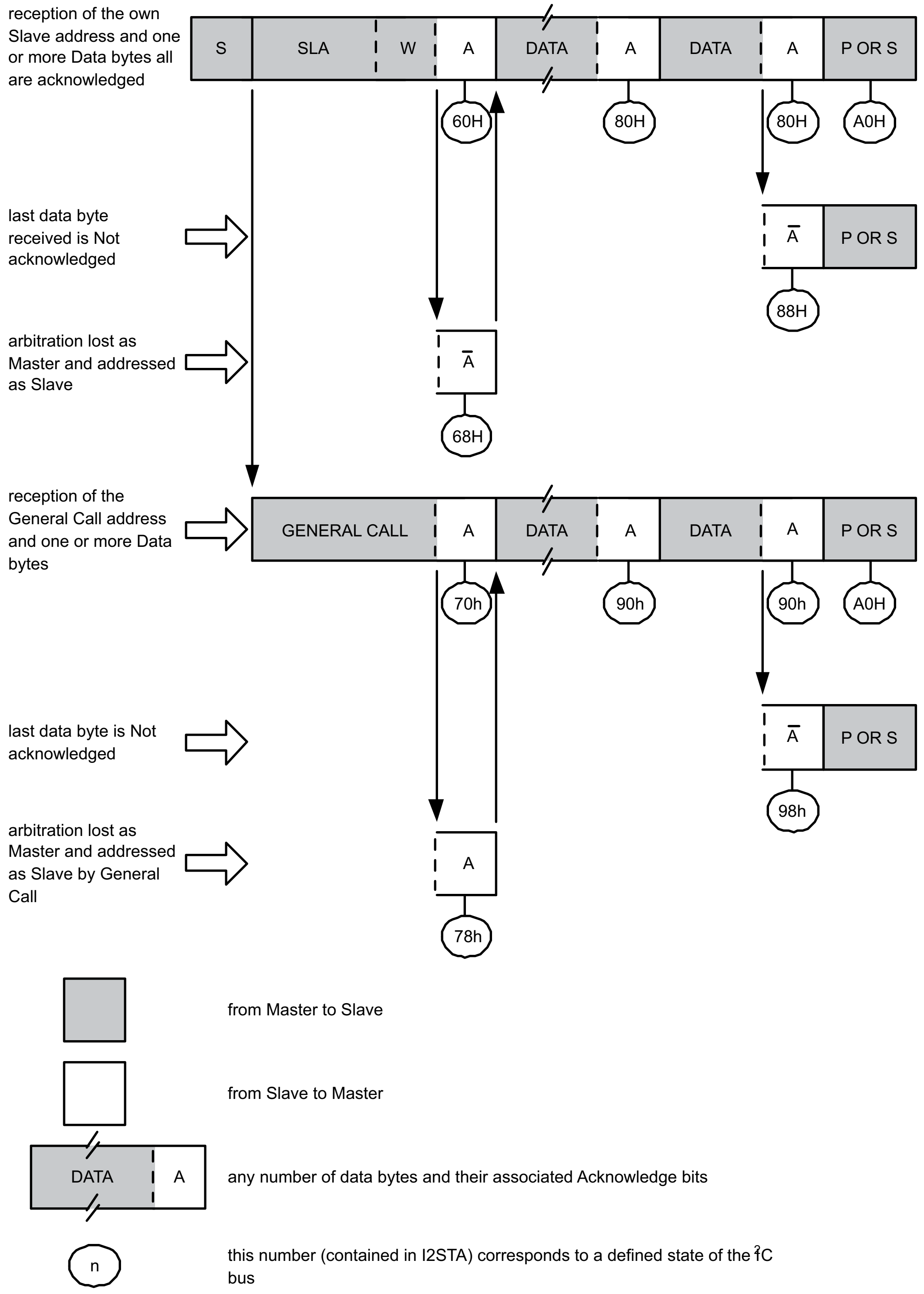


Fig 39. Format and states in the Slave Receiver mode

13.7.1.4 Slave Transmitter mode

In the slave transmitter mode, a number of data bytes are transmitted to a master receiver (see Figure 40). Data transfer is initialized as in the slave receiver mode. When ADR and CON have been initialized, the I^2C block waits until it is addressed by its own slave address followed by the data direction bit which must be “1” (R) for the I^2C block to operate in the slave transmitter mode. After its own slave address and the R bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from STAT. This status code is used to vector to a state service routine, and the appropriate action to be taken for each of these status codes is detailed in Table 215. The slave transmitter mode may also be entered if arbitration is lost while the I^2C block is in the master mode (see state 0xB0).

If the AA bit is reset during a transfer, the I^2C block will transmit the last byte of the transfer and enter state 0xC0 or 0xC8. The I^2C block is switched to the not addressed slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all 1s as serial data. While AA is reset, the I^2C block does not respond to its own slave address or a General Call address. However, the I^2C-bus is still monitored, and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I^2C block from the I^2C-bus.

Table 215. Slave Transmitter mode

Status Code (STAT)	Status of the I^{2}C-bus and hardware	Application software response					Next action taken by I^{2}C hardware
		To/From DAT	To CON				
			STA	STO	SI	AA	
0xA8	Own SLA+R has been received; ACK has been returned.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received.
0xB0	Arbitration lost in SLA+R/W as master; Own SLA+R has been received, ACK has been returned.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xB8	Data byte in DAT has been transmitted; ACK has been received.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xC0	Data byte in DAT has been transmitted; NOT ACK has been received.	No DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		No DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if ADR[0] = logic 1.
		No DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		No DAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0xC8	Last data byte in DAT has been transmitted (AA = 0); ACK has been received.	No DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		No DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if ADR[0] = logic 1.
		No DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		No DAT action	1	0	0	01	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.

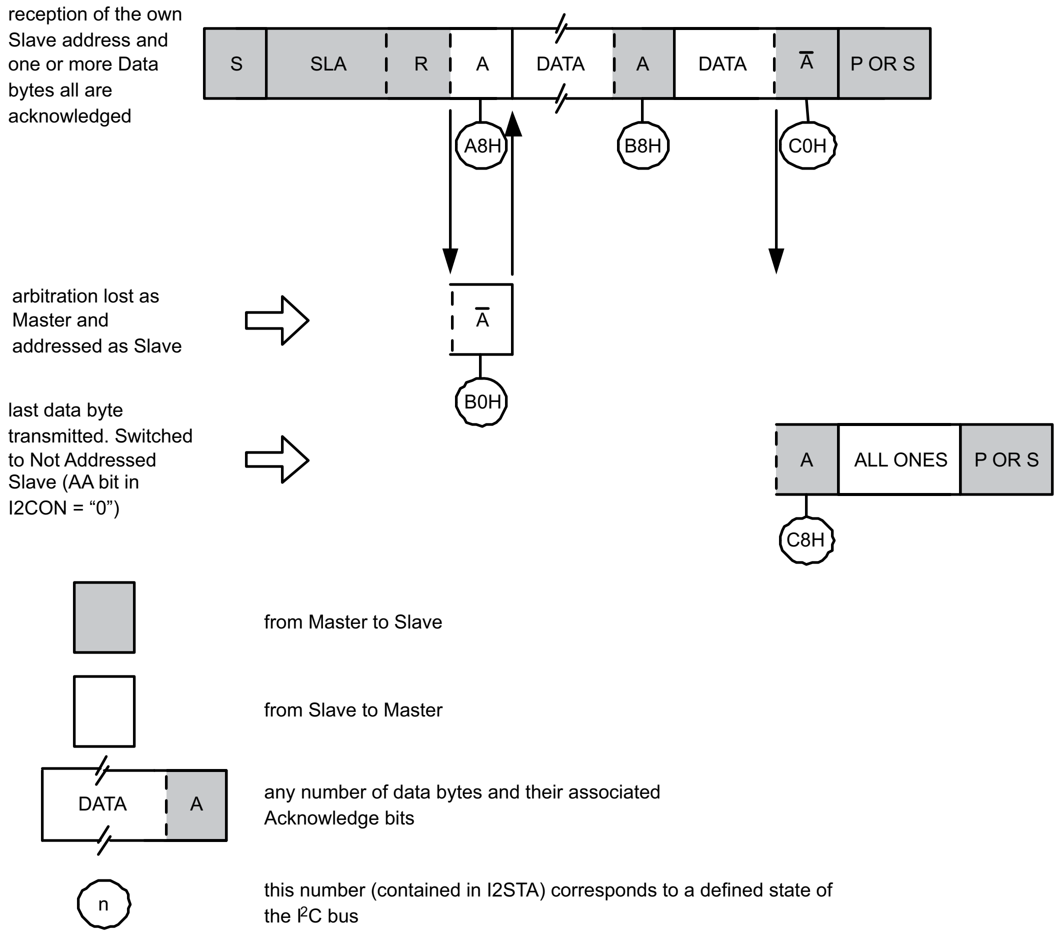


Fig 40. Format and states in the Slave Transmitter mode

13.7.1.5 Miscellaneous states

There are two STAT codes that do not correspond to a defined I^{2}C hardware state (see Table 216). These are discussed below.

13.7.1.5.1 STAT = 0xF8

This status code indicates that no relevant information is available because the serial interrupt flag, SI, is not yet set. This occurs between other states and when the I^{2}C block is not involved in a serial transfer.

13.7.1.5.2 STAT = 0x00

This status code indicates that a bus error has occurred during an I^{2}C serial transfer. A bus error is caused when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. A bus error may also be caused when external interference disturbs the internal I^{2}C block signals. When a bus error occurs, SI is set. To recover from a bus error, the STO flag must be set and SI must be cleared. This causes the I²C block to enter the “not addressed” slave mode (a defined state) and to clear the STO flag (no other bits in CON are affected). The SDA and SCL lines are released (a STOP condition is not transmitted).

Table 216. Miscellaneous States

Status Code (STAT)	Status of the I²C-bus and hardware	Application software response					Next action taken by I²C hardware
		To/From DAT	To CON				
			STA	STO	SI	AA	
0xF8	No relevant state information available; SI = 0.	No DAT action	No CON action				Wait or proceed current transfer.
0x00	Bus error during MST or selected slave modes, due to an illegal START or STOP condition. State 0x00 can also occur when interference causes the I²C block to enter an undefined state.	No DAT action	0	1	0	X	Only the internal hardware is affected in the MST or addressed SLV modes. In all cases, the bus is released and the I²C block is switched to the not addressed SLV mode. STO is reset.

13.7.1.6 Some special cases

The I²C hardware has facilities to handle the following special cases that may occur during a serial transfer:

- Simultaneous Repeated START conditions from two masters
- Data transfer after loss of arbitration
- Forced access to the I²C-bus
- I²C-bus obstructed by a LOW level on SCL or SDA
- Bus error

13.7.1.6.1 Simultaneous Repeated START conditions from two masters

A Repeated START condition may be generated in the master transmitter or master receiver modes. A special case occurs if another master simultaneously generates a Repeated START condition (see Figure 41). Until this occurs, arbitration is not lost by either master since they were both transmitting the same data.

If the I²C hardware detects a Repeated START condition on the I²C-bus before generating a Repeated START condition itself, it will release the bus, and no interrupt request is generated. If another master frees the bus by generating a STOP condition, the I²C block will transmit a normal START condition (state 0x08), and a retry of the total serial data transfer can commence.

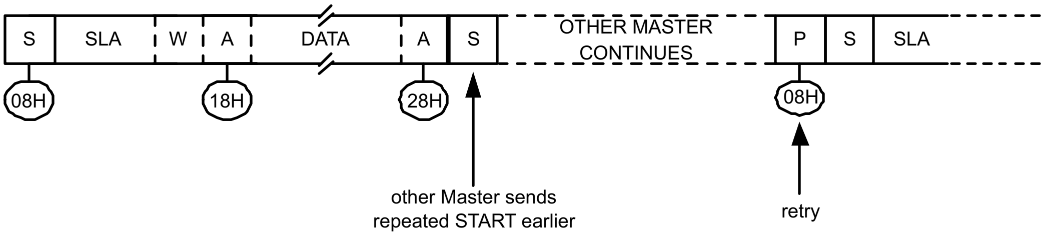


Fig 41. Simultaneous Repeated START conditions from two masters

13.7.1.6.2 Data transfer after loss of arbitration

Arbitration may be lost in the master transmitter and master receiver modes (see Figure 29). Loss of arbitration is indicated by the following states in STAT; 0x38, 0x68, 0x78, and 0xB0 (see Figure 37 and Figure 38).

If the STA flag in CON is set by the routines which service these states, then, if the bus is free again, a START condition (state 0x08) is transmitted without intervention by the CPU, and a retry of the total serial transfer can commence.

13.7.1.6.3 Forced access to the I^{2}C-bus

In some applications, it may be possible for an uncontrolled source to cause a bus hang-up. In such situations, the problem may be caused by interference, temporary interruption of the bus or a temporary short-circuit between SDA and SCL.

If an uncontrolled source generates a superfluous START or masks a STOP condition, then the I^{2}C-bus stays busy indefinitely. If the STA flag is set and bus access is not obtained within a reasonable amount of time, then a forced access to the I^{2}C-bus is possible. This is achieved by setting the STO flag while the STA flag is still set. No STOP condition is transmitted. The I^{2}C hardware behaves as if a STOP condition was received and is able to transmit a START condition. The STO flag is cleared by hardware (see Figure 42).

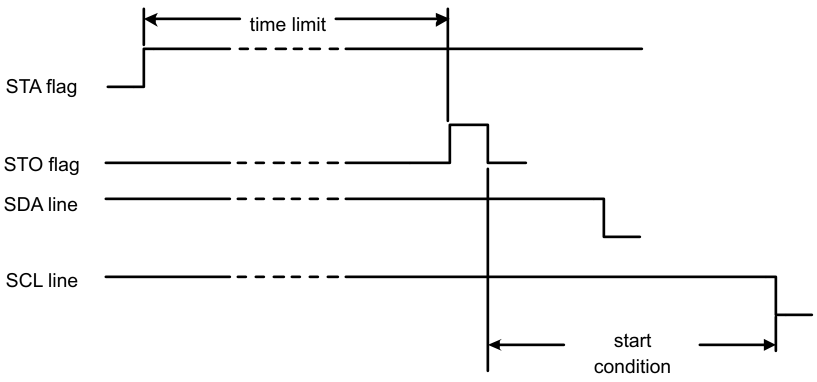
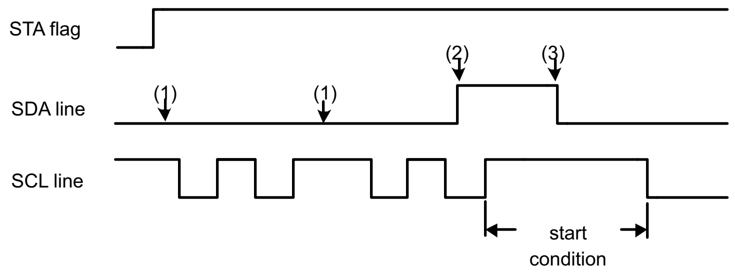


Fig 42. Forced access to a busy I^{2}C-bus

13.7.1.6.4 I^{2}C-bus obstructed by a LOW level on SCL or SDA

An I^{2}C-bus hang-up can occur if either the SDA or SCL line is held LOW by any device on the bus. If the SCL line is obstructed (pulled LOW) by a device on the bus, no further serial transfer is possible, and the problem must be resolved by the device that is pulling the SCL bus line LOW.

Typically, the SDA line may be obstructed by another device on the bus that has become out of synchronization with the current bus master by either missing a clock, or by sensing a noise pulse as a clock. In this case, the problem can be solved by transmitting additional clock pulses on the SCL line (see Figure 43). The I^{2}C interface does not include a dedicated time-out timer to detect an obstructed bus, but this can be implemented using another timer in the system. When detected, software can force clocks (up to 9 may be required) on SCL until SDA is released by the offending device. At that point, the slave may still be out of synchronization, so a START should be generated to insure that all I^{2}C peripherals are synchronized.



(1) Unsuccessful attempt to send a START condition.
(2) SDA line is released.
(3) Successful attempt to send a START condition. State 08H is entered.

Fig 43. Recovering from a bus obstruction caused by a LOW level on SDA

13.7.1.6.5 Bus error

A bus error occurs when a START or STOP condition is detected at an illegal position in the format frame. Examples of illegal positions are during the serial transfer of an address byte, a data bit, or an acknowledge bit.

The I^{2}C hardware only reacts to a bus error when it is involved in a serial transfer either as a master or an addressed slave. When a bus error is detected, the I^{2}C block immediately switches to the not addressed slave mode, releases the SDA and SCL lines, sets the interrupt flag, and loads the status register with 0x00. This status code may be used to vector to a state service routine which either attempts the aborted serial transfer again or simply recovers from the error condition as shown in Table 216.

13.7.1.7 I^{2}C state service routines

This section provides examples of operations that must be performed by various I^{2}C state service routines. This includes:

- Initialization of the I^{2}C block after a Reset.
- I^{2}C Interrupt Service
- The 26 state service routines providing support for all four I^{2}C operating modes.

13.7.1.8 Initialization

In the initialization example, the I^2C block is enabled for both master and slave modes. For each mode, a buffer is used for transmission and reception. The initialization routine performs the following functions:

- ADR is loaded with the part's own slave address and the General Call bit (GC)
- The I^2C interrupt enable and interrupt priority bits are set
- The slave mode is enabled by simultaneously setting the I2EN and AA bits in CON and the serial clock frequency (for master modes) is defined by is defined by loading the SCLH and SCLL registers. The master routines must be started in the main program.

The I^2C hardware now begins checking the I^2C-bus for its own slave address and General Call. If the General Call or the own slave address is detected, an interrupt is requested and STAT is loaded with the appropriate state information.

13.7.1.9 I^2C interrupt service

When the I^2C interrupt is entered, STAT contains a status code which identifies one of the 26 state services to be executed.

13.7.1.10 The state service routines

Each state routine is part of the I^2C interrupt routine and handles one of the 26 states.

13.7.1.11 Adapting state services to an application

The state service examples show the typical actions that must be performed in response to the 26 I^2C state codes. If one or more of the four I^2C operating modes are not used, the associated state services can be omitted, as long as care is taken that the those states can never occur.

In an application, it may be desirable to implement some kind of time-out during I^2C operations, in order to trap an inoperative bus or a lost service routine.

13.7.2 Software example

13.7.2.1 Initialization routine

Example to initialize I^2C Interface as a Slave and/or Master.

1. Load ADR with own Slave Address, enable General Call recognition if needed.
2. Enable I^2C interrupt.
3. Write 0x44 to CONSET to set the I2EN and AA bits, enabling Slave functions. For Master only functions, write 0x40 to CONSET.

13.7.2.2 Start Master Transmit function

Begin a Master Transmit operation by setting up the buffer, pointer, and data count, then initiating a START.

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Write bit.
3. Write 0x20 to CONSET to set the STA bit.

4. Set up data to be transmitted in Master Transmit buffer.
5. Initialize the Master data counter to match the length of the message being sent.
6. Exit

13.7.2.3 Start Master Receive function

Begin a Master Receive operation by setting up the buffer, pointer, and data count, then initiating a START.

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Read bit.
3. Write 0x20 to CONSET to set the STA bit.
4. Set up the Master Receive buffer.
5. Initialize the Master data counter to match the length of the message to be received.
6. Exit

13.7.2.4 I^{2}C interrupt routine

Determine the I^{2}C state and which state routine will be used to handle it.

1. Read the I^{2}C status from STA.
2. Use the status value to branch to one of 26 possible state routines.

13.7.2.5 Non mode specific states

13.7.2.5.1 State: 0x00

Bus Error. Enter not addressed Slave mode and release bus.

1. Write 0x14 to CONSET to set the STO and AA bits.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

13.7.2.5.2 Master States

State 08 and State 10 are for both Master Transmit and Master Receive modes. The R/W bit decides whether the next state is within Master Transmit mode or Master Receive mode.

13.7.2.5.3 State: 0x08

A START condition has been transmitted. The Slave Address + R/W bit will be transmitted, an ACK bit will be received.

1. Write Slave Address with R/W bit to DAT.
2. Write 0x04 to CONSET to set the AA bit.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Set up Master Transmit mode data buffer.
5. Set up Master Receive mode data buffer.
6. Initialize Master data counter.
7. Exit

13.7.2.5.4 State: 0x10

A Repeated START condition has been transmitted. The Slave Address + R/W bit will be transmitted, an ACK bit will be received.

1. Write Slave Address with R/W bit to DAT.
2. Write 0x04 to CONSET to set the AA bit.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Set up Master Transmit mode data buffer.
5. Set up Master Receive mode data buffer.
6. Initialize Master data counter.
7. Exit

13.7.2.6 Master Transmitter states

13.7.2.6.1 State: 0x18

Previous state was State 8 or State 10, Slave Address + Write has been transmitted, ACK has been received. The first data byte will be transmitted, an ACK bit will be received.

1. Load DAT with first data byte from Master Transmit buffer.
2. Write 0x04 to CONSET to set the AA bit.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Increment Master Transmit buffer pointer.
5. Exit

13.7.2.6.2 State: 0x20

Slave Address + Write has been transmitted, NOT ACK has been received. A STOP condition will be transmitted.

1. Write 0x14 to CONSET to set the STO and AA bits.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

13.7.2.6.3 State: 0x28

Data has been transmitted, ACK has been received. If the transmitted data was the last data byte then transmit a STOP condition, otherwise transmit the next data byte.

1. Decrement the Master data counter, skip to step 5 if not the last data byte.
2. Write 0x14 to CONSET to set the STO and AA bits.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Exit
5. Load DAT with next data byte from Master Transmit buffer.
6. Write 0x04 to CONSET to set the AA bit.
7. Write 0x08 to CONCLR to clear the SI flag.
8. Increment Master Transmit buffer pointer
9. Exit

13.7.2.6.4 State: 0x30

Data has been transmitted, NOT ACK received. A STOP condition will be transmitted.

1. Write 0x14 to CONSET to set the STO and AA bits.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

13.7.2.6.5 State: 0x38

Arbitration has been lost during Slave Address + Write or data. The bus has been released and not addressed Slave mode is entered. A new START condition will be transmitted when the bus is free again.

1. Write 0x24 to CONSET to set the STA and AA bits.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

13.7.2.7 Master Receive states

13.7.2.7.1 State: 0x40

Previous state was State 08 or State 10. Slave Address + Read has been transmitted, ACK has been received. Data will be received and ACK returned.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

13.7.2.7.2 State: 0x48

Slave Address + Read has been transmitted, NOT ACK has been received. A STOP condition will be transmitted.

1. Write 0x14 to CONSET to set the STO and AA bits.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

13.7.2.7.3 State: 0x50

Data has been received, ACK has been returned. Data will be read from DAT. Additional data will be received. If this is the last data byte then NOT ACK will be returned, otherwise ACK will be returned.

1. Read data byte from DAT into Master Receive buffer.
2. Decrement the Master data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to CONCLR to clear the SI flag and the AA bit.
4. Exit
5. Write 0x04 to CONSET to set the AA bit.
6. Write 0x08 to CONCLR to clear the SI flag.
7. Increment Master Receive buffer pointer
8. Exit

13.7.2.7.4 State: 0x58

Data has been received, NOT ACK has been returned. Data will be read from DAT. A STOP condition will be transmitted.

1. Read data byte from DAT into Master Receive buffer.
2. Write 0x14 to CONSET to set the STO and AA bits.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Exit

13.7.2.8 Slave Receiver states

13.7.2.8.1 State: 0x60

Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK returned.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

13.7.2.8.2 State: 0x68

Arbitration has been lost in Slave Address and R/W bit as bus Master. Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK will be returned. STA is set to restart Master mode after the bus is free again.

1. Write 0x24 to CONSET to set the STA and AA bits.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit.

13.7.2.8.3 State: 0x70

General call has been received, ACK has been returned. Data will be received and ACK returned.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

13.7.2.8.4 State: 0x78

Arbitration has been lost in Slave Address + R/W bit as bus Master. General call has been received and ACK has been returned. Data will be received and ACK returned. STA is set to restart Master mode after the bus is free again.

1. Write 0x24 to CONSET to set the STA and AA bits.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

13.7.2.8.5 State: 0x80

Previously addressed with own Slave Address. Data has been received and ACK has been returned. Additional data will be read.

1. Read data byte from DAT into the Slave Receive buffer.
2. Decrement the Slave data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to CONCLR to clear the SI flag and the AA bit.
4. Exit.
5. Write 0x04 to CONSET to set the AA bit.
6. Write 0x08 to CONCLR to clear the SI flag.
7. Increment Slave Receive buffer pointer.
8. Exit

13.7.2.8.6 State: 0x88

Previously addressed with own Slave Address. Data has been received and NOT ACK has been returned. Received data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

13.7.2.8.7 State: 0x90

Previously addressed with General Call. Data has been received, ACK has been returned. Received data will be saved. Only the first data byte will be received with ACK. Additional data will be received with NOT ACK.

1. Read data byte from DAT into the Slave Receive buffer.
2. Write 0x0C to CONCLR to clear the SI flag and the AA bit.
3. Exit

13.7.2.8.8 State: 0x98

Previously addressed with General Call. Data has been received, NOT ACK has been returned. Received data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

13.7.2.8.9 State: 0xA0

A STOP condition or Repeated START has been received, while still addressed as a Slave. Data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

13.7.2.9 Slave Transmitter states

13.7.2.9.1 State: 0xA8

Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received.

1. Load DAT from Slave Transmit buffer with first data byte.
2. Write 0x04 to CONSET to set the AA bit.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

13.7.2.9.2 State: 0xB0

Arbitration lost in Slave Address and R/W bit as bus Master. Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received. STA is set to restart Master mode after the bus is free again.

1. Load DAT from Slave Transmit buffer with first data byte.
2. Write 0x24 to CONSET to set the STA and AA bits.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

13.7.2.9.3 State: 0xB8

Data has been transmitted, ACK has been received. Data will be transmitted, ACK bit will be received.

1. Load DAT from Slave Transmit buffer with data byte.
2. Write 0x04 to CONSET to set the AA bit.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Increment Slave Transmit buffer pointer.
5. Exit

13.7.2.9.4 State: 0xC0

Data has been transmitted, NOT ACK has been received. Not addressed Slave mode is entered.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit.

13.7.2.9.5 State: 0xC8

The last data byte has been transmitted, ACK has been received. Not addressed Slave mode is entered.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

14.1 How to read this chapter

SSP0 and SSP1 are available on all parts.

14.2 Features

- Compatible with Motorola SPI, 4-wire TI SSI, and National Semiconductor Microwire buses.
- Synchronous Serial Communication.
- Supports master or slave operation.
- Eight-frame FIFOs for both transmit and receive.
- 4-bit to 16-bit frame.

14.3 Basic configuration

1. Pins: The SSP/SPI pins must be configured in the IOCON register block.
2. Power: In the SYSAHBCLKCTRL register, set bit 11 for SSP0 and bit 18 for SSP1 (Table 40).
3. Peripheral clock: Enable the SSP0/SSP1 peripheral clocks by writing to the SSP0/1CLKDIV registers (Table 41/Table 43).
4. Reset: Before accessing the SSP/SPI block, ensure that the SSP0/1_RST_N bits (bit 0 and bit 2) in the PRESETCTRL register (Table 23) are set to 1. This de-asserts the reset signal to the SSP/SPI block.

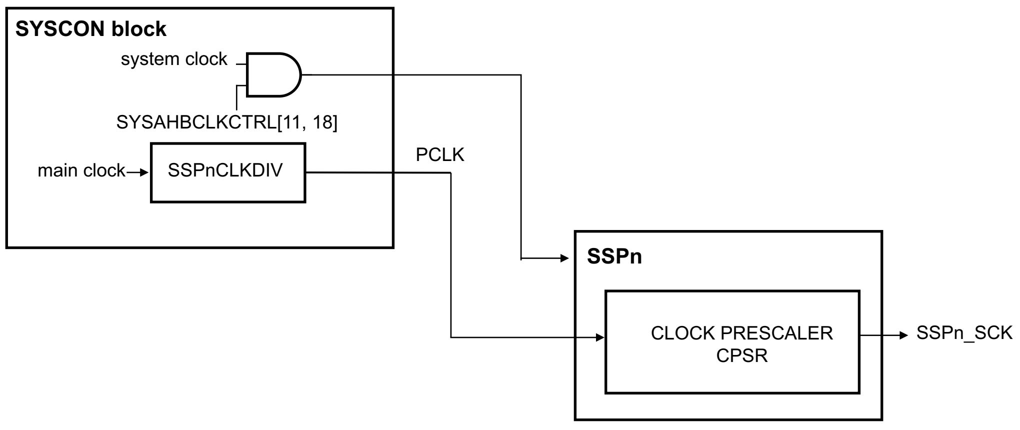


Fig 44. SSP0/1 clocking

14.4 General description

The SSP/SPI is a Synchronous Serial Port (SSP) controller capable of operation on a SPI, 4-wire SSI, or Microwire bus. It can interact with multiple masters and slaves on the bus. Only a single master and a single slave can communicate on the bus during a given data transfer. Data transfers are in principle full duplex, with frames of 4 bits to 16 bits of data flowing from the master to the slave and from the slave to the master. In practice it is often the case that only one of these data flows carries meaningful data.

14.5 Pin description

Table 217. SSP/SPI pin descriptions

Pin name	Type	Interface pin name/function			Pin description
		SPI	SSI	Microwire	
SSP0_SCK, SSP1_SCK	I/O	SCK	CLK	SK	**Serial Clock.** SCK/CLK/SK is a clock signal used to synchronize the transfer of data. It is driven by the master and received by the slave. When SSP/SPI interface is used, the clock is programmable to be active-high or active-low, otherwise it is always active-high. SCK only switches during a data transfer. Any other time, the SSP/SPI interface either holds it in its inactive state or does not drive it (leaves it in high-impedance state).
SSP0_SSEL, SSP1_SSEL	I/O	SSEL	FS	CS	**Frame Sync/Slave Select.** When the SSP/SPI interface is a bus master, it drives this signal to an active state before the start of serial data and then releases it to an inactive state after the data has been sent.The active state of this signal can be high or low depending upon the selected bus and mode. When the SSP/SPI interface is a bus slave, this signal qualifies the presence of data from the Master according to the protocol in use. When there is just one bus master and one bus slave, the Frame Sync or Slave Select signal from the Master can be connected directly to the slave’s corresponding input. When there is more than one slave on the bus, further qualification of their Frame Select/Slave Select inputs will typically be necessary to prevent more than one slave from responding to a transfer.
SSP0_MISO, SSP1_MISO	I/O	MISO	DR(M) DX(S)	SI(M) SO(S)	**Master In Slave Out.** The MISO signal transfers serial data from the slave to the master. When the SSP/SPI is a slave, serial data is output on this signal. When the SSP/SPI is a master, it clocks in serial data from this signal. When the SSP/SPI is a slave and is not selected by FS/SSEL, it does not drive this signal (leaves it in high-impedance state).
SSP0_MOSI, SSP1_MOSI	I/O	MOSI	DX(M) DR(S)	SO(M) SI(S)	**Master Out Slave In.** The MOSI signal transfers serial data from the master to the slave. When the SSP/SPI is a master, it outputs serial data on this signal. When the SSP/SPI is a slave, it clocks in serial data from this signal.

14.6 Register description

The register addresses of the SSP controllers are shown in Table 218.

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

Table 218. Register overview: SSP/SPI0 (base address 0x4004 0000)

Name	Access	Address offset	Description	Reset value	Reference
CR0	R/W	0x000	Control Register 0. Selects the serial clock rate, bus type, and data size.	0	Table 220
CR1	R/W	0x004	Control Register 1. Selects master/slave and other modes.	0	Table 221
DR	R/W	0x008	Data Register. Writes fill the transmit FIFO, and reads empty the receive FIFO.	0	Table 222
SR	RO	0x00C	Status Register	0x0000 0003	Table 223
CPSR	R/W	0x010	Clock Prescale Register	0	Table 224
IMSC	R/W	0x014	Interrupt Mask Set and Clear Register	0	Table 225
RIS	RO	0x018	Raw Interrupt Status Register	0x0000 0008	Table 226
MIS	RO	0x01C	Masked Interrupt Status Register	0	Table 227
ICR	WO	0x020	SSPICR Interrupt Clear Register	NA	Table 228
DMACR	R/W	0x024	DMA control register	0	Table 229

Table 219. Register overview: SSP/SPI1 (base address 0x4005 8000)

Name	Access	Address offset	Description	Reset value	Reference
CR0	R/W	0x000	Control Register 0. Selects the serial clock rate, bus type, and data size.	0	Table 220
CR1	R/W	0x004	Control Register 1. Selects master/slave and other modes.	0	Table 221
DR	R/W	0x008	Data Register. Writes fill the transmit FIFO, and reads empty the receive FIFO.	0	Table 222
SR	RO	0x00C	Status Register	0x0000 0003	Table 223
CPSR	R/W	0x010	Clock Prescale Register	0	Table 224
IMSC	R/W	0x014	Interrupt Mask Set and Clear Register	0	Table 225
RIS	RO	0x018	Raw Interrupt Status Register	0x0000 0008	Table 226
MIS	RO	0x01C	Masked Interrupt Status Register	0	Table 227
ICR	WO	0x020	SSPICR Interrupt Clear Register	NA	Table 228
DMACR	R/W	0x024	DMA control register	0	Table 229

14.6.1 SSP/SPI Control Register 0

This register controls the basic operation of the SSP/SPI controller.

Table 220. SSP/SPI Control Register 0 (CR0, address 0x4004 0000 (SSP0) and 0x4005 8000 (SSP1)) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	DSS		Data Size Select. This field controls the number of bits transferred in each frame. Values 0000-0010 are not supported and should not be used.	0000
		0x3	4-bit transfer	
		0x4	5-bit transfer	
		0x5	6-bit transfer	
		0x6	7-bit transfer	
		0x7	8-bit transfer	
		0x8	9-bit transfer	
		0x9	10-bit transfer	
		0xA	11-bit transfer	
		0xB	12-bit transfer	
		0xC	13-bit transfer	
		0xD	14-bit transfer	
		0xE	15-bit transfer	
		0xF	16-bit transfer	
5:4	FRF		Frame Format.	00
		0x0	SPI	
		0x1	TI	
		0x2	Microwire	
		0x3	This combination is not supported and should not be used.	
6	CPOL		Clock Out Polarity. This bit is only used in SPI mode.	0
		0	SPI controller maintains the bus clock low between frames.	
		1	SPI controller maintains the bus clock high between frames.	
7	CPHA		Clock Out Phase. This bit is only used in SPI mode.	0
		0	SPI controller captures serial data on the first clock transition of the frame, that is, the transition **away from** the inter-frame state of the clock line.	
		1	SPI controller captures serial data on the second clock transition of the frame, that is, the transition **back to** the inter-frame state of the clock line.	
15:8	SCR		Serial Clock Rate. The number of prescaler output clocks per bit on the bus, minus one. Given that CPSDVSR is the prescale divider, and the APB clock PCLK clocks the prescaler, the bit frequency is PCLK / (CPSDVSR × [SCR+1]).	0x00
31:16	-	-	Reserved	-

14.6.2 SSP/SPI Control Register 1

This register controls certain aspects of the operation of the SSP/SPI controller.

Table 221. SSP/SPI Control Register 1 (CR1, address 0x4004 0004 (SSP0) and 0x4005 8004 (SSP1)) bit description

Bit	Symbol	Value	Description	Reset Value
0	LBM		Loop Back Mode.	0
		0	During normal operation.	
		1	Serial input is taken from the serial output (MOSI or MISO) rather than the serial input pin (MISO or MOSI respectively).	
1	SSE		SPI Enable.	0
		0	The SPI controller is disabled.	
		1	The SPI controller will interact with other devices on the serial bus. Software should write the appropriate control information to the other SSP/SPI registers and interrupt controller registers, before setting this bit.	
2	MS		Master/Slave Mode.This bit can only be written when the SSE bit is 0.	0
		0	The SPI controller acts as a master on the bus, driving the SCLK, MOSI, and SSEL lines and receiving the MISO line.	
		1	The SPI controller acts as a slave on the bus, driving MISO line and receiving SCLK, MOSI, and SSEL lines.	
3	SOD		Slave Output Disable. This bit is relevant only in slave mode (MS = 1). If it is 1, this blocks this SPI controller from driving the transmit data line (MISO).	0
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.6.3 SSP/SPI Data Register

Software can write data to be transmitted to this register and read data that has been received.

Table 222. SSP/SPI Data Register (DR, address 0x4004 0008 (SSP0) and 0x4005 8008 (SSP1)) bit description

Bit	Symbol	Description	Reset Value
15:0	DATA	**Write:** software can write data to be sent in a future frame to this register whenever the TNF bit in the Status register is 1, indicating that the Tx FIFO is not full. If the Tx FIFO was previously empty and the SPI controller is not busy on the bus, transmission of the data will begin immediately. Otherwise the data written to this register will be sent as soon as all previous data has been sent (and received). If the data length is less than 16 bit, software must right-justify the data written to this register. **Read:** software can read data from this register whenever the RNE bit in the Status register is 1, indicating that the Rx FIFO is not empty. When software reads this register, the SPI controller returns data from the least recent frame in the Rx FIFO. If the data length is less than 16 bit, the data is right-justified in this field with higher order bits filled with 0s.	0x0000
31:16	-	Reserved.	-

14.6.4 SSP/SPI Status Register

This read-only register reflects the current status of the SPI controller.

Table 223. SSP/SPI Status Register (SR, address 0x4004 000C (SSP0) and 0x4005 800C (SSP1)) bit description

Bit	Symbol	Description	Reset Value
0	TFE	Transmit FIFO Empty. This bit is 1 is the Transmit FIFO is empty, 0 if not.	1
1	TNF	Transmit FIFO Not Full. This bit is 0 if the Tx FIFO is full, 1 if not.	1
2	RNE	Receive FIFO Not Empty. This bit is 0 if the Receive FIFO is empty, 1 if not.	0
3	RFF	Receive FIFO Full. This bit is 1 if the Receive FIFO is full, 0 if not.	0
4	BSY	Busy. This bit is 0 if the SPI controller is idle, 1 if it is currently sending/receiving a frame and/or the Tx FIFO is not empty.	0
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.6.5 SSP/SPI Clock Prescale Register

This register controls the factor by which the Prescaler divides the SPI peripheral clock SPI_PCLK to yield the prescaler clock that is, in turn, divided by the SCR factor in the SSPCR0 registers, to determine the bit clock.

Table 224. SSP/SPI Clock Prescale Register (CPSR, address 0x4004 0010 (SSP0) and 0x4005 8010 (SSP1)) bit description

Bit	Symbol	Description	Reset Value
7:0	CPSDVSR	This even value between 2 and 254, by which SPI_PCLK is divided to yield the prescaler output clock. Bit 0 always reads as 0.	0
31:8	-	Reserved.	-

Important: the CPSR value must be properly initialized, or the SPI controller will not be able to transmit data correctly.

In Slave mode, the SPI clock rate provided by the master must not exceed 1/12 of the SPI peripheral clock selected. The content of the SSPnCPSR register is not relevant.

In master mode, $\text{CPSDVSR}_{\text{min}}$ = 2 or larger (even numbers only).

14.6.6 SSP/SPI Interrupt Mask Set/Clear Register

This register controls whether each of the four possible interrupt conditions in the SPI controller are enabled.

Table 225. SSP/SPI Interrupt Mask Set/Clear register (IMSC, address 0x4004 0014 (SSP0) and 0x4005 8014 (SSP1)) bit description

Bit	Symbol	Description	Reset Value
0	RORIM	Software should set this bit to enable interrupt when a Receive Overrun occurs, that is, when the Rx FIFO is full and another frame is completely received. The ARM spec implies that the preceding frame data is overwritten by the new frame data when this occurs.	0
1	RTIM	Software should set this bit to enable interrupt when a Receive Time-out condition occurs. A Receive Time-out occurs when the Rx FIFO is not empty, and no has not been read for a time-out period. The time-out period is the same for master and slave modes and is determined by the SSP bit rate: 32 bits at PCLK / (CPSDVSR × [SCR+1]).	0
2	RXIM	Software should set this bit to enable interrupt when the Rx FIFO is at least half full.	0
3	TXIM	Software should set this bit to enable interrupt when the Tx FIFO is at least half empty.	0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.6.7 SSP/SPI Raw Interrupt Status Register

This read-only register contains a 1 for each interrupt condition that is asserted, regardless of whether or not the interrupt is enabled in the IMSC registers.

Table 226. SSP/SPI Raw Interrupt Status register (RIS, address 0x4004 0018 (SSP0) and 0x4005 8018 (SSP1)) bit description

	Symbol	Description	Reset value
0	RORRIS	This bit is 1 if another frame was completely received while the RxFIFO was full. The ARM spec implies that the preceding frame data is overwritten by the new frame data when this occurs.	0
1	RTRIS	This bit is 1 if the Rx FIFO is not empty, and has not been read for a time-out period. The time-out period is the same for master and slave modes and is determined by the SSP bit rate: 32 bits at PCLK / (CPSDVSR × [SCR+1]).	0
2	RXRIS	This bit is 1 if the Rx FIFO is at least half full.	0
3	TXRIS	This bit is 1 if the Tx FIFO is at least half empty.	1
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.6.8 SSP/SPI Masked Interrupt Status Register

This read-only register contains a 1 for each interrupt condition that is asserted and enabled in the IMSC registers. When an SSP/SPI interrupt occurs, the interrupt service routine should read this register to determine the causes of the interrupt.

Table 227. SSP/SPI Masked Interrupt Status register (MIS, address 0x4004 001C (SSP0) and 0x4005 801C (SSP1)) bit description

Bit	Symbol	Description	Reset value
0	RORMIS	This bit is 1 if another frame was completely received while the RxFIFO was full, and this interrupt is enabled.	0
1	RTMIS	This bit is 1 if the Rx FIFO is not empty, has not been read for a time-out period, and this interrupt is enabled. The time-out period is the same for master and slave modes and is determined by the SSP bit rate: 32 bits at PCLK / (CPSDVSR × [SCR+1]).	0
2	RXMIS	This bit is 1 if the Rx FIFO is at least half full, and this interrupt is enabled.	0
3	TXMIS	This bit is 1 if the Tx FIFO is at least half empty, and this interrupt is enabled.	0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.6.9 SSP/SPI Interrupt Clear Register

Software can write one or more ones to this write-only register, to clear the corresponding interrupt conditions in the SPI controller. Note that the other two interrupt conditions can be cleared by writing or reading the appropriate FIFO or disabled by clearing the corresponding bit in SSPIMSC registers.

Table 228. SSP/SPI interrupt Clear Register (ICR, address 0x4004 0020 (SSP0) and 0x4005 8020 (SSP1)) bit description

Bit	Symbol	Description	Reset Value
0	RORIC	Writing a 1 to this bit clears the “frame was received when RxFIFO was full” interrupt.	NA
1	RTIC	Writing a 1 to this bit clears the Rx FIFO was not empty and has not been read for a timeout period interrupt. The timeout period is the same for master and slave modes and is determined by the SSP bit rate: 32 bits at PCLK / (CPSDVSR × [SCR+1]).	NA
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.6.10 SSP/SPI DMA Control Register

The DMACR register is the DMA control register. It is a read/write register.

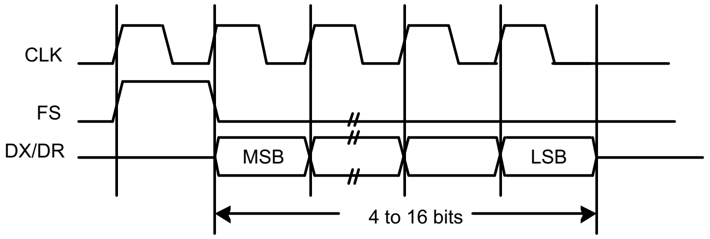
Table 229: SSP/SPI DMA Control Register (DMACR, address 0x4004 0024 (SSP0) and 0x4005 8024 (SSP1)) bit description

Bit	Symbol	Description	Reset value
0	RXDMAE	Receive DMA Enable. When this bit is set to one 1, DMA for the receive FIFO is enabled, otherwise receive DMA is disabled.	0
1	TXDMAE	Transmit DMA Enable. When this bit is set to one 1, DMA for the transmit FIFO is enabled, otherwise transmit DMA is disabled	0
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

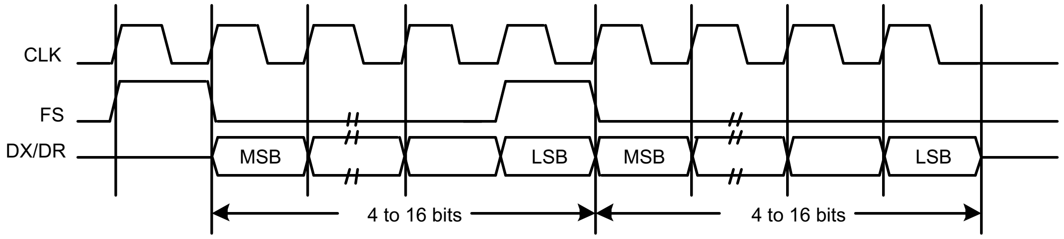
14.7 Functional description

14.7.1 Texas Instruments synchronous serial frame format

Figure 45 shows the 4-wire Texas Instruments synchronous serial frame format supported by the SPI module.



a. Single frame transfer



b. Continuous/back-to-back frames transfer

Fig 45. Texas Instruments Synchronous Serial Frame Format: a) Single and b) Continuous/back-to-back two frames transfer

For device configured as a master in this mode, CLK and FS are forced LOW, and the transmit data line DX is in 3-state mode whenever the SSP is idle. Once the bottom entry of the transmit FIFO contains data, FS is pulsed HIGH for one CLK period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of CLK, the MSB of the 4-bit to 16-bit data frame is shifted out on the DX pin. Likewise, the MSB of the received data is shifted onto the DR pin by the off-chip serial slave device.

Both the SSP and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each CLK. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of CLK after the LSB has been latched.

14.7.2 SPI frame format

The SPI interface is a four-wire interface where the SSEL signal behaves as a slave select. The main feature of the SPI format is that the inactive state and phase of the SCK signal are programmable through the CPOL and CPHA bits within the SSPCR0 control register.

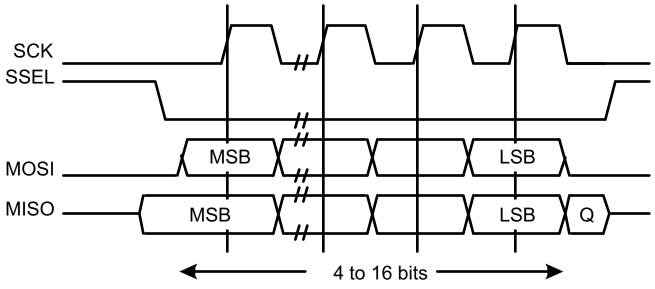
14.7.2.1 Clock Polarity (CPOL) and Phase (CPHA) control

When the CPOL clock polarity control bit is LOW, it produces a steady state low value on the SCK pin. If the CPOL clock polarity control bit is HIGH, a steady state high value is placed on the CLK pin when data is not being transferred.

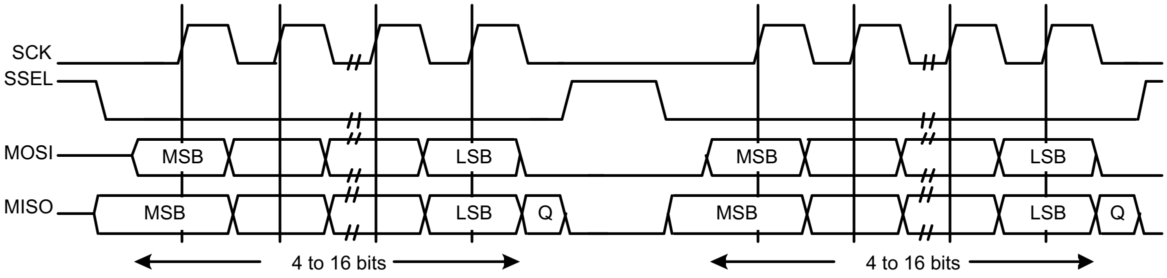
The CPHA control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the CPHA phase control bit is LOW, data is captured on the first clock edge transition. If the CPHA clock phase control bit is HIGH, data is captured on the second clock edge transition.

14.7.2.2 SPI format with CPOL=0,CPHA=0

Single and continuous transmission signal sequences for SPI format with CPOL = 0, CPHA = 0 are shown in Figure 46.



a. Single transfer with CPOL=0 and CPHA=0



b. Continuous transfer with CPOL=0 and CPHA=0

Fig 46. SPI frame format with CPOL=0 and CPHA=0 (a) single and b) continuous transfer

In this configuration, during idle periods:

- The CLK signal is forced LOW.
- SSEL is forced HIGH.
- The transmit MOSI/MISO pad is in high impedance.

If the SSP/SPI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW. This causes slave data to be enabled onto the MISO input line of the master. Master's MOSI is enabled.

One half SCK period later, valid master data is transferred to the MOSI pin. Now that both the master and slave data have been set, the SCK master clock pin goes HIGH after one further half SCK period.

The data is captured on the rising and propagated on the falling edges of the SCK signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSEL signal must be pulsed HIGH between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the CPHA bit is logic zero. Therefore the master device must raise the SSEL pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSEL pin is returned to its idle state one SCK period after the last bit has been captured.

14.7.2.3 SPI format with CPOL=0,CPHA=1

The transfer signal sequence for SPI format with CPOL = 0, CPHA = 1 is shown in Figure 47, which covers both single and continuous transfers.

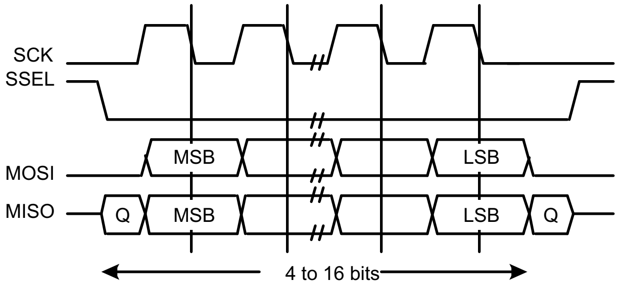


Fig 47. SPI frame format with CPOL=0 and CPHA=1

In this configuration, during idle periods:

- The CLK signal is forced LOW.
- SSEL is forced HIGH.
- The transmit MOSI/MISO pad is in high impedance.

If the SSP/SPI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW. Master's MOSI pin is enabled. After a further one half SCK period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the SCK is enabled with a rising edge transition.

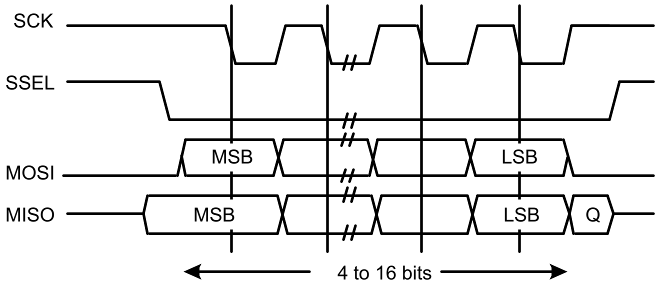
Data is then captured on the falling edges and propagated on the rising edges of the SCK signal.

In the case of a single word transfer, after all bits have been transferred, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured.

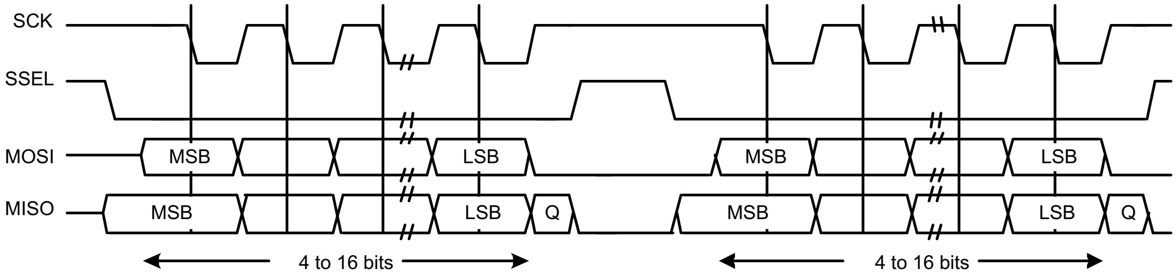
For continuous back-to-back transfers, the SSEL pin is held LOW between successive data words and termination is the same as that of the single word transfer.

14.7.2.4 SPI format with CPOL = 1,CPHA = 0

Single and continuous transmission signal sequences for SPI format with CPOL=1, CPHA=0 are shown in Figure 48.



a. Single transfer with CPOL=1 and CPHA=0



b. Continuous transfer with CPOL=1 and CPHA=0

Fig 48. SPI frame format with CPOL = 1 and CPHA = 0 (a) Single and b) Continuous Transfer)

In this configuration, during idle periods:

- The CLK signal is forced HIGH.
- SSEL is forced HIGH.
- The transmit MOSI/MISO pad is in high impedance.

If the SSP/SPI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW, which causes slave data to be immediately transferred onto the MISO line of the master. Master’s MOSI pin is enabled.

One half period later, valid master data is transferred to the MOSI line. Now that both the master and slave data have been set, the SCK master clock pin becomes LOW after one further half SCK period. This means that data is captured on the falling edges and be propagated on the rising edges of the SCK signal.

In the case of a single word transmission, after all bits of the data word are transferred, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSEL signal must be pulsed HIGH between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the CPHA bit is logic zero. Therefore the master device must raise the SSEL pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSEL pin is returned to its idle state one SCK period after the last bit has been captured.

14.7.2.5 SPI format with CPOL = 1,CPHA = 1

The transfer signal sequence for SPI format with CPOL = 1, CPHA = 1 is shown in Figure 49, which covers both single and continuous transfers.

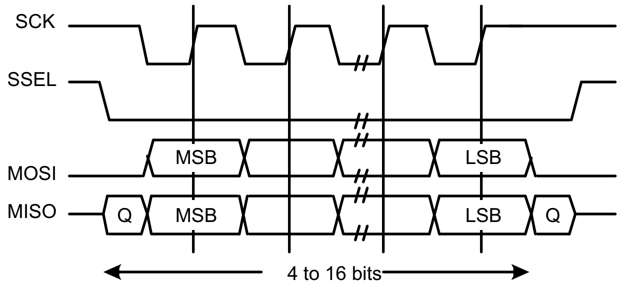


Fig 49. SPI Frame Format with CPOL = 1 and CPHA = 1

In this configuration, during idle periods:

- The CLK signal is forced HIGH.
- SSEL is forced HIGH.
- The transmit MOSI/MISO pad is in high impedance.

If the SSP/SPI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW. Master’s MOSI is enabled. After a further one half SCK period, both master and slave data are enabled onto their respective transmission lines. At the same time, the SCK is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SCK signal.

After all bits have been transferred, in the case of a single word transmission, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured. For continuous back-to-back transmissions, the SSEL pins remains in its active LOW state, until the final bit of the last word has been captured, and then returns to its idle state as described above. In general, for continuous back-to-back transfers the SSEL pin is held LOW between successive data words and termination is the same as that of the single word transfer.

14.7.3 Semiconductor Microwire frame format

Figure 50 shows the Microwire frame format for a single frame. Figure 51 shows the same format when back-to-back frames are transmitted.

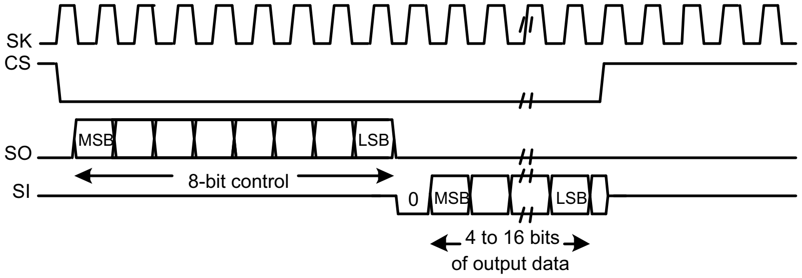


Fig 50. Microwire frame format (single transfer)

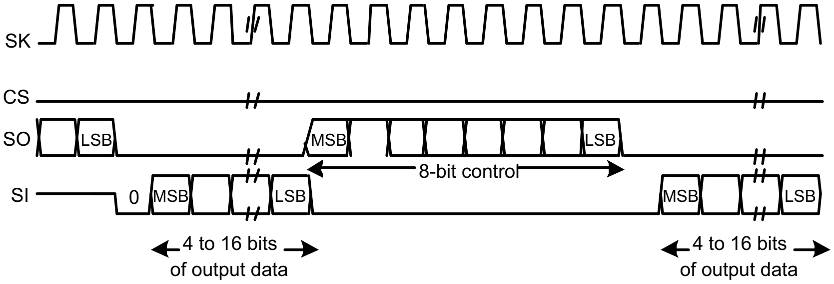


Fig 51. Microwire frame format (continuous transfers)

Microwire format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSP/SPI to the off-chip slave device. During this transmission, no incoming data is received by the SSP/SPI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bit in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

- The SK signal is forced LOW.
- CS is forced HIGH.
- The transmit data line SO is arbitrarily forced LOW.

A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of CS causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto the SO pin. CS remains LOW for the duration of the frame transmission. The SI pin remains tri-stated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each SK. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSP/SPI. Each bit is driven onto SI line on the falling edge of SK. The SSP/SPI in turn latches each bit on the rising edge of SK. At the end of the frame, for single transfers, the CS signal is pulled HIGH one clock period after the last bit has been latched in the receive serial shifter, that causes the data to be transferred to the receive FIFO.

Note: The off-chip slave device can tri-state the receive line either on the falling edge of SK after the LSB has been latched by the receive shiftier, or when the CS pin goes HIGH.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the CS line is continuously asserted (held LOW) and transmission of data occurs back to back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge SK, after the LSB of the frame has been latched into the SSP/SPI.

14.7.3.1 Setup and hold time requirements on CS with respect to SK in Microwire mode

In the Microwire mode, the SSP/SPI slave samples the first bit of receive data on the rising edge of SK after CS has gone LOW. Masters that drive a free-running SK must ensure that the CS signal has sufficient setup and hold margins with respect to the rising edge of SK.

Figure 52 illustrates these setup and hold time requirements. With respect to the SK rising edge on which the first bit of receive data is to be sampled by the SSP/SPI slave, CS must have a setup of at least two times the period of SK on which the SSP/SPI operates. With respect to the SK rising edge previous to this edge, CS must have a hold of at least one SK period.

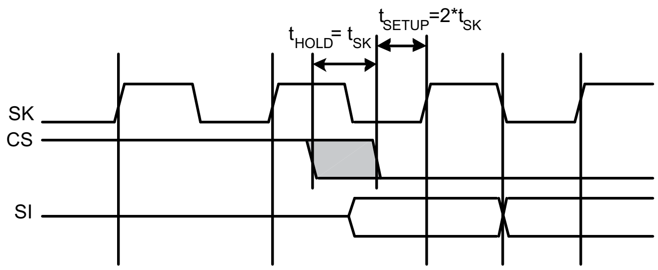


Fig 52. Microwire frame format setup and hold details

UM10732

Chapter 15: LPC11U6x USB2.0 full-speed device controller

Rev. 1.8 — 19 December 2016 **User manual**

15.1 How to read this chapter

The USB controller is available on the LPC11U6x only.

15.2 Features

- USB2.0 full-speed device controller.
- Supports 10 physical (5 logical) endpoints including one control endpoint.
- Single and double-buffering supported.
- Each non-control endpoint supports bulk, interrupt, or isochronous endpoint types.
- Supports wake-up from Deep-sleep mode on USB activity and remote wake-up.
- Supports SoftConnect through internal 1.5 kΩ pull-up resistor between USB_DP and V_{DD}.
- Link Power Management (LPM) supported.
- USB pads include internal softconnect and 33 Ω series termination resistor for USB_DP and USB_DM signal lines.
- Support for XTAL-less low-speed USB.

15.3 Basic configuration

- Pins: Configure the USB pins in the IOCON register block.
- In the SYSAHBCLKCTRL register, enable the clock to the USB controller register interface by setting bit 14 and to the USB RAM by setting bit 27 (see Table 40).
- Power: Enable the power to the USB PHY and to the USB PLL, if used, in the PDRUNCFG register (Table 69).
- Configure the USB clock divider (see Table 45 and Table 47).
- Configure the USB wake-up signal (see Section 15.3.1) if needed.

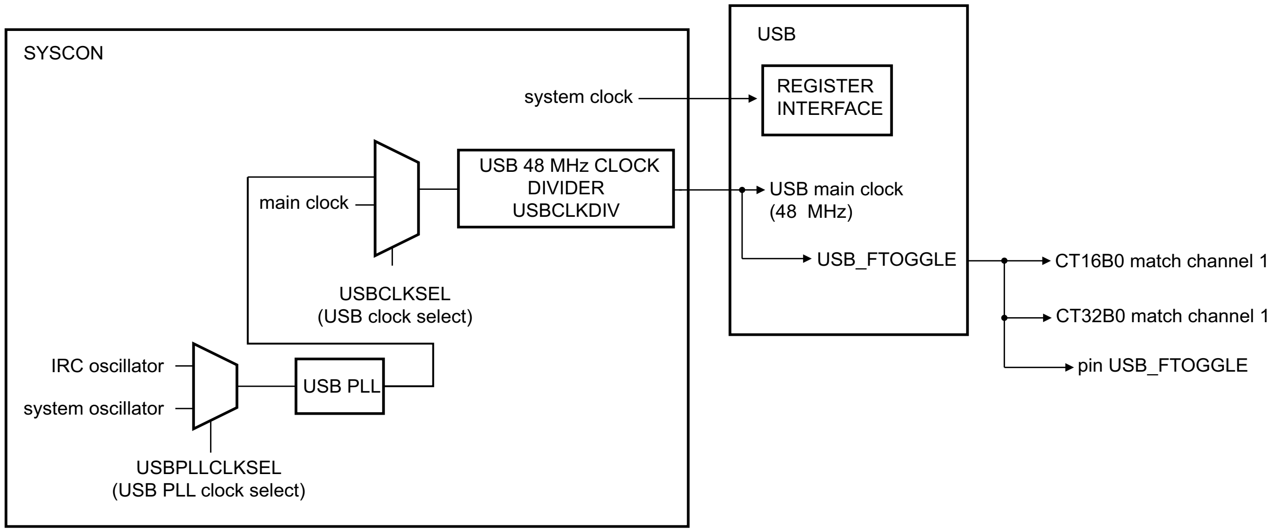


Fig 53. USB clocking

15.3.1 USB wake-up

15.3.1.1 Waking up from Deep-sleep and Power-down modes on USB activity

To allow the part to wake up from Deep-sleep or Power-down mode on USB activity, complete the following steps:

1. Set bit AP_CLK in the USBCLKCTRL register (Table 63) to 0 (default) to enable automatic control of the USB need_clock signal.
2. Wait until USB activity is suspended by polling the DSUS bit in the DSVCMD_STAT register (DSUS = 1).
3. The USB need_clock signal will be deasserted after another 2 ms. Poll the USBCLKST register until the USB need_clock status bit is 0 (Table 64).
4. Once the USBCLKST register returns 0, enable the USB activity wake-up interrupt in the NVIC (# 30) and clear it.
5. Set bit 1 in the USBCLKCTRL register to 1 to trigger the USB activity wake-up interrupt on the rising edge of the USB need_clock signal.
6. Enable the wake-up from Deep-sleep or Power-down modes on this interrupt by enabling the USB need_clock signal in the STARTERP1 register (Table 66, bit 19).
7. Enter Deep-sleep or Power-down modes by writing to the PCON register.
8. Execute a WFI instruction.

The part will automatically wake up and resume execution on USB activity.

15.3.1.2 Remote wake-up

To issue a remote wake-up when the USB activity is suspended, complete the following steps:

1. Set bit AP_CLK in the USBCLKCTRL register to 0 (Table 63, default) to enable automatic control of the USB need_clock signal.

2. When it is time to issue a remote wake-up, turn on the USB clock and enable the USB clock source.
3. Force the USB clock on by writing a 1 to bit AP_CLK (Table 63, bit 0) in the USBCLKCTRL register.
4. Write a 0 to the DSUS bit in the DSVCMD_STAT register.
5. Wait until the USB leaves the suspend state by polling the DSUS bit in the DSVCMD_STAT register (DSUS =0).
6. Clear the AP_CLK bit (Table 63, bit 0) in the USBCLKCTRL to enable automatic USB clock control.

15.4 General description

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

The host schedules transactions in 1 ms frames. Each frame contains a Start-Of-Frame (SOF) marker and transactions that transfer data to or from device endpoints. Each device can have a maximum of 16 logical or 32 physical endpoints. The device controller supports up to 10 physical endpoints. There are four types of transfers defined for the endpoints. Control transfers are used to configure the device.

Interrupt transfers are used for periodic data transfer. Bulk transfers are used when the latency of transfer is not critical. Isochronous transfers have guaranteed delivery time but no error correction.

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

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

Figure 54 shows the block diagram of the USB device controller.

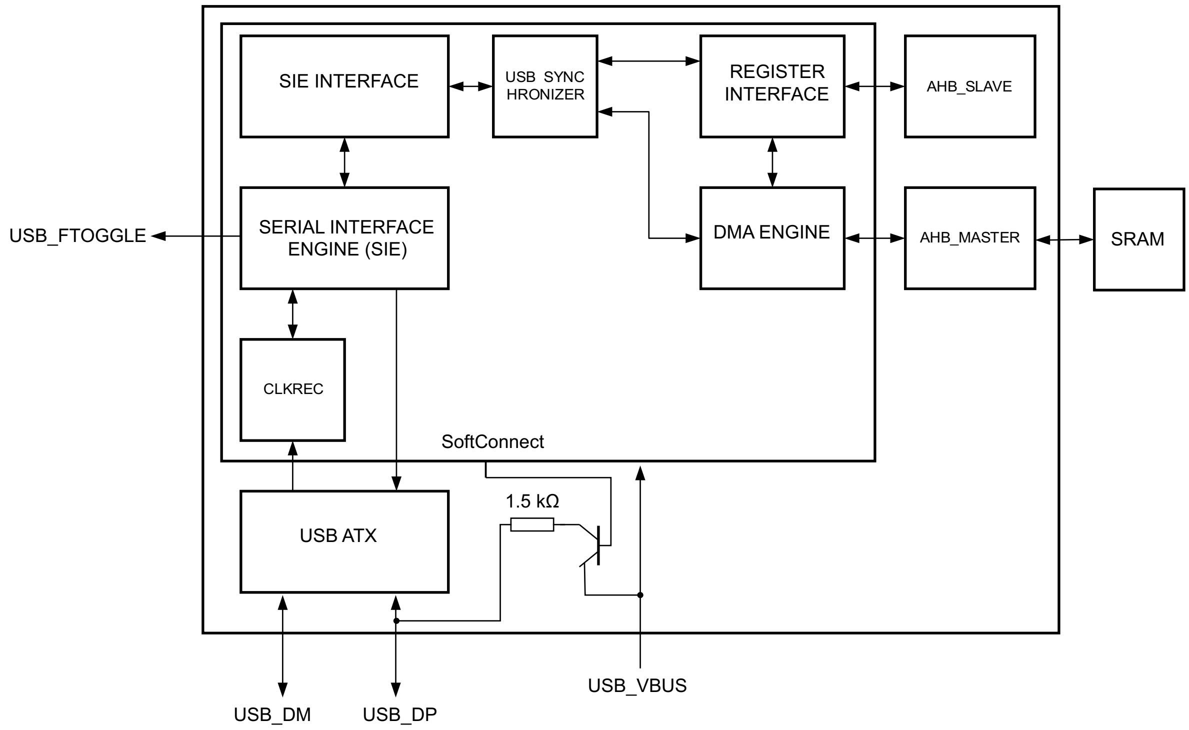


Fig 54. USB block diagram

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

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

15.4.1 USB software interface

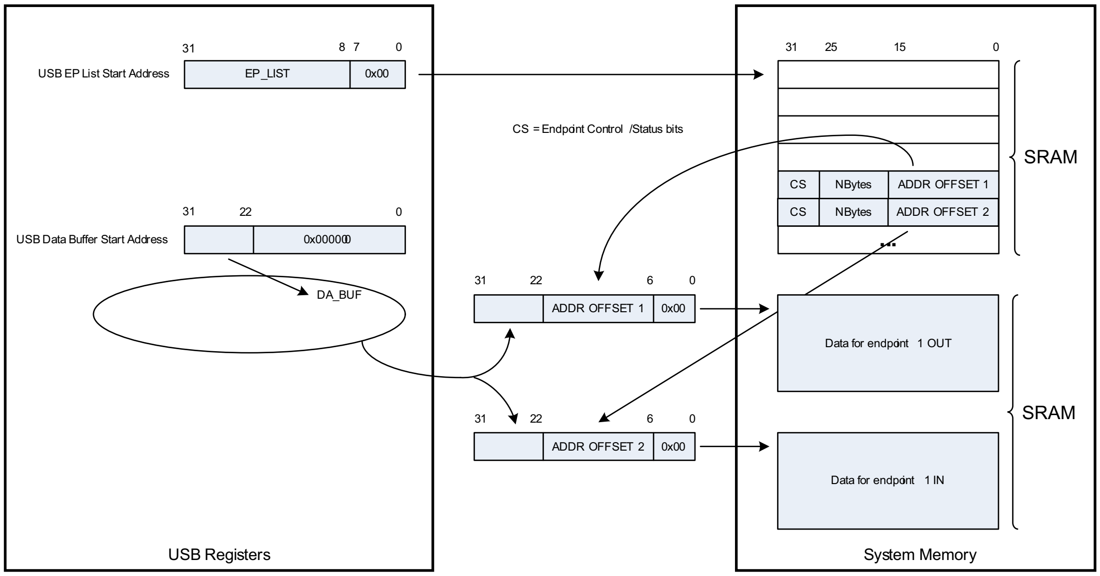


Fig 55. USB software interface

15.4.2 Fixed endpoint configuration

Table 230 shows the supported endpoint configurations. The packet size is configurable up to the maximum value shown in Table 230 for each type of end point.

Table 230. Fixed endpoint configuration

Logical endpoint	Physical endpoint	Endpoint type	Direction	Max packet size (byte)	Double buffer
0	0	Control	Out	64	No
0	1	Control	In	64	No
1	2	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
1	3	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes
2	4	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
2	5	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes
3	6	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
3	7	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes
4	8	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
4	9	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes

15.4.3 SoftConnect

Software can control the USB_CONNECT signal by setting the DCON bit in the DEVCMDSTAT register. If the DCON bit is set to 1, the USB_DP line is pulled up to V_{DD} through an internal 1.5 KOhm pull-up resistor.

The purpose of the soft connect feature using USB_CONNECT is to control when the device connects to the bus. When the device detects a USB_VBUS signal on the bus, it can finish processing if necessary, and then under software control indicate its presence to the host by pulling the USB_DP line HIGH.In a similar way, software can re-initialize a USB connection without the necessity to unplug the USB cable.

15.4.4 Interrupts

The USB controller has two interrupt lines USB_Int_Req_IRQ and USB_Int_Req_FIQ. Software can program the corresponding bit in the USB interrupt routing register to route the interrupt condition to one of these entries in the NVIC table Table 6. An interrupt is generated by the hardware if both the interrupt status bit and the corresponding interrupt enable bit are set. The interrupt status bit is set by hardware if the interrupt condition occurs (irrespective of the interrupt enable bit setting).

15.4.5 Suspend and resume

The USB protocol insists on power management by the USB device. This becomes even more important if the device draws power from the bus (bus-powered device). The following constraints should be met by the bus-powered device.

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

A device will go into the L2 suspend state if there is no activity on the USB bus for more than 3 ms. A suspended device wakes up, if there is transmission from the host (host-initiated wake up). The USB controller also supports software initiated remote wake-up. To initiate remote wake-up, software on the device must enable all clocks and clear the suspend bit. This will cause the hardware to generate a remote wake-up signal upstream.

The USB controller supports Link Power Management (LPM). Link Power Management defines an additional link power management state L1 that supplements the existing L2 state by utilizing most of the existing suspend/resume infrastructure but provides much faster transitional latencies between L1 and L0 (On).

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

If there is no activity for the next 2 ms, the USB need_clock signal will go low. This indicates that the USB main clock can be switched off.

When activity is detected on the USB bus, the USB suspend signal is deactivated and USB need_clock signal is activated. This process is fully combinatorial and hence no USB main clock is required to activate the US B need_clock signal.

15.4.6 Frame toggle output

The USB_FTOGGLE output pin reflects the 1 kHz clock derived from the incoming Start of Frame tokens sent by the USB host. When the USB is connected to a host, the rising edge of the USB_FTOGGLE signal is aligned with the middle of the SOF token which is received on the USB bus. The signal can be monitored on a pin (connected through the IOCON) or on the capture inputs(CAP1) of timers CT16B0 or CT32B0.

When no tokens are coming in, the USB_FTOGGLE input is a 1 KHz signal based on the USB main clock.

15.4.7 Clocking

The USB device controller has the following clock connections:

- USB main clock: The USB main clock is the 48 MHz +/- 500 ppm clock from the dedicated USB PLL or the main clock (see Table 45). If the main clock is used, the system PLL output must be 48 MHz. The clock source for the USB PLL or the system PLL must be derived from the system oscillator if the USB is operated in full-speed mode. For low-speed mode, the IRC is suitable as the clock source.

  The USB main clock is used to recover the 12 MHz clock from the USB bus.
- AHB clock: This is the AHB system bus clock. The minimum frequency of the AHB clock is 16 MHz when the USB device controller is receiving or transmitting USB packets.

15.4.8 USB Low-speed operation

The USB device controller can be used in low-speed mode supporting 1.5 Mbit/s data exchange with a USB host controller.

Remark: To operate in low-speed mode, change the board connections as follows:

1. Connect USB_DP to the D- pin of the connector.
2. Connect USB_DM to the D+ pin of the connector.

To configure the USB clock for low-speed USB, follow these steps:

1. Select the IRC as clock source for the USB PLL. See Table 35.
2. Configure the USB to generate a 48 MHz clock.
3. Divide the 48 MHz clock by 8 to obtain a 6 MHz low-speed USB clock. See Table 47.

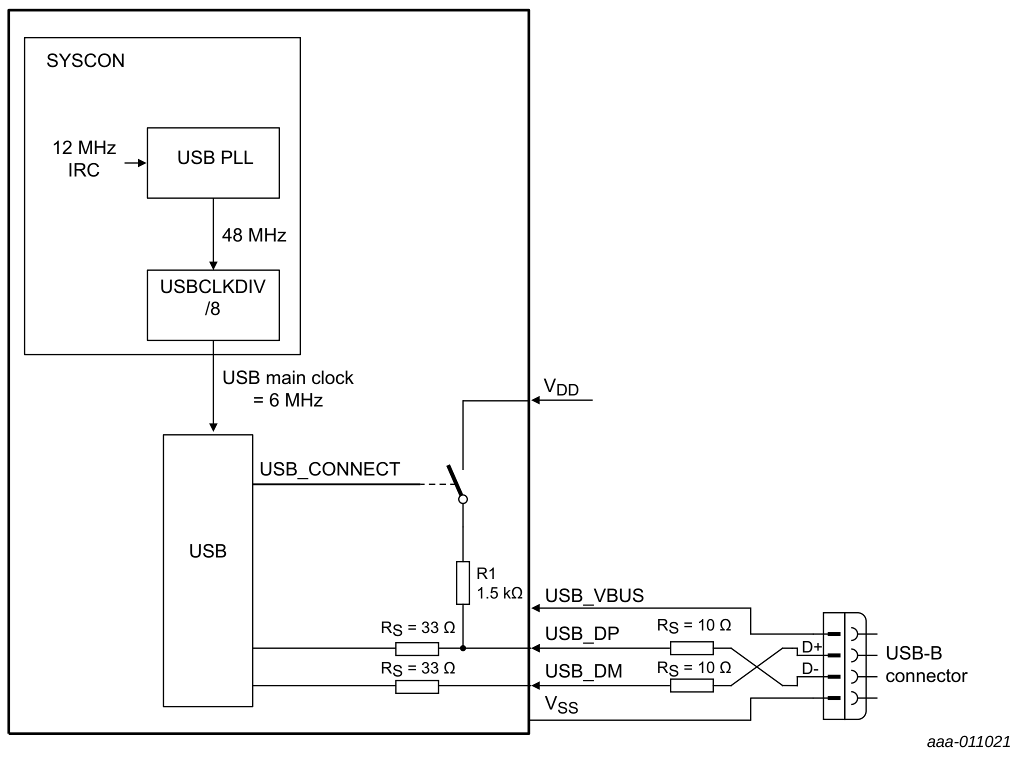


Fig 56. USB Low-speed XTAL-less connections

15.5 Pin description

The device controller can access one USB port.

Table 231. USB device pin description

Name	Direction	Description
V_{BUS}	I	V_{BUS} status input. When this function is not enabled via its corresponding IOCON register, it is driven HIGH internally.
USB_FTOGGLE	O	USB 1 ms SoF signal.
USB_DP	I/O	Positive differential data. Includes internal 33 Ω series resistor.
USB_DM	I/O	Negative differential data. Includes internal 33 Ω series resistor.

15.6 Register description

Table 232. Register overview: USB (base address: 0x4008 0000)

Name	Access	Address offset	Description	Reset value	Reference
DEVCMDSTAT	R/W	0x000	USB Device Command/Status register	0x00000 800	Table 233
INFO	R/W	0x004	USB Info register	0	Table 234
EPLISTSTART	R/W	0x008	USB EP Command/Status List start address	0	Table 235
DATABUFSTART	R/W	0x00C	USB Data buffer start address	0	Table 236
LPM	R/W	0x010	USB Link Power Management register	0	Table 237
EPSKIP	R/W	0x014	USB Endpoint skip	0	Table 238
EPINUSE	R/W	0x018	USB Endpoint Buffer in use	0	Table 239
EPBUFCFG	R/W	0x01C	USB Endpoint Buffer Configuration register	0	Table 240
INTSTAT	R/W	0x020	USB interrupt status register	0	Table 241
INTEN	R/W	0x024	USB interrupt enable register	0	Table 242
INTSETSTAT	R/W	0x028	USB set interrupt status register	0	Table 243
INTROUTING	R/W	0x02C	USB interrupt routing register	0	Table 244
EPTOGGLE	R	0x034	USB Endpoint toggle register	0	Table 245

15.6.1 USB Device Command/Status register

Table 233. USB Device Command/Status register (DEVCMDSTAT, address 0x4008 0000) bit description

Bit	Symbol	Value	Description	Reset value	Access
6:0	DEV_ADDR		USB device address. After bus reset, the address is reset to 0x00. If the enable bit is set, the device will respond on packets for function address DEV_ADDR. When receiving a SetAddress Control Request from the USB host, software must program the new address before completing the status phase of the SetAddress Control Request.	0	RW
7	DEV_EN		USB device enable. If this bit is set, the HW will start responding on packets for function address DEV_ADDR.	0	RW
8	SETUP		SETUP token received. If a SETUP token is received and acknowledged by the device, this bit is set. As long as this bit is set all received IN and OUT tokens will be NAKed by HW. SW must clear this bit by writing a one. If this bit is zero, HW will handle the tokens to the CTRL EP0 as indicated by the CTRL EP0 IN and OUT data information programmed by SW.	0	RWC
9	PLL_ON		Always PLL Clock on:	0	RW
		0	Functional. USB_NeedClk functional.		
		1	High. USB_NeedClk always 1. Clock will not be stopped in case of suspend.		
10	-		Reserved.	0	RO

Table 233. USB Device Command/Status register (DEVCMDSTAT, address 0x4008 0000) bit description

Bit	Symbol	Value	Description	Reset value	Access
11	LPM_SUP		LPM Support.:	1	RW
		0	No. LPM not supported.		
		1	Yes.LPM supported.		
12	INTONNAK_AO		Interrupt on NAK for interrupt and bulk OUT EP	0	RW
		0	AK only. Only acknowledged packets generate an interrupt		
		1	Ak and Nak. Both acknowledged and NAKed packets generate interrupts.		
13	INTONNAK_AI		Interrupt on NAK for interrupt and bulk IN EP	0	RW
		0	AK only. Only acknowledged packets generate an interrupt		
		1	Ak and NAK. Both acknowledged and NAKed packets generate interrupts.		
14	INTONNAK_CO		Interrupt on NAK for control OUT EP	0	RW
		0	AK only. Only acknowledged packets generate an interrupt		
		1	AK and NAK. Both acknowledged and NAKed packets generate interrupts.		
15	INTONNAK_CI		Interrupt on NAK for control IN EP	0	RW
		0	AK only. Only acknowledged packets generate an interrupt		
		1	AK and NAK. Both acknowledged and NAKed packets generate interrupts.		
16	DCON		Device status - connect.	0	RW
		0	Not connected.		
		1	Connect. The connect bit must be set by software to indicate that the device must signal a connect. The pull-up resistor on USB_DP will be enabled when this bit is set and the VBUSDEBOUNCED bit is one.		
17	DSUS		Device status - suspend. The suspend bit indicates the current suspend state. It is set to 1 when the device hasn't seen any activity on its upstream port for more than 3 milliseconds. It is reset to 0 on any activity. When the device is suspended (Suspend bit DSUS = 1) and the software writes a 0 to it, the device will generate a remote wake-up. This will only happen when the device is connected (Connect bit = 1). When the device is not connected or not suspended, a writing a 0 has no effect. Writing a 1 never has an effect.	0	RW
18	-		Reserved.	0	RO

Table 233. USB Device Command/Status register (DEVCMDSTAT, address 0x4008 0000) bit description

Bit	Symbol	Value	Description	Reset value	Access
19	LPM_SUS		Device status - LPM Suspend. This bit represents the current LPM suspend state. It is set to 1 by HW when the device has acknowledged the LPM request from the USB host and the Token Retry Time of 10 μs has elapsed. When the device is in the LPM suspended state (LPM suspend bit = 1) and the software writes a zero to this bit, the device will generate a remote walk-up. Software can only write a zero to this bit when the LPM_REWP bit is set to 1. HW resets this bit when it receives a host initiated resume. HW only updates the LPM_SUS bit when the LPM_SUPP bit is equal to one.	0	RW
20	LPM_REWP		LPM Remote Wake-up Enabled by USB host. HW sets this bit to one when the BREMOTEWAKE bit in the LPM extended token is set to 1. HW will reset this bit to 0 when it receives the host initiated LPM resume, when a remote wake-up is sent by the device or when a USB bus reset is received. Software can use this bit to check if the remote wake-up feature is enabled by the host for the LPM transaction.	0	RO
23:20	-		Reserved.	0	RO
24	DCON_C		Device status - connect change. The Connect Change bit is set when the device's pull-up resistor is disconnected because VBus disappeared. The bit is reset by writing a one to it.	0	RWC
25	DSUS_C		Device status - suspend change. The suspend change bit is set to 1 when the suspend bit toggles. The suspend bit can toggle because: - The device goes in the suspended state - The device is disconnected - The device receives resume signaling on its upstream port. The bit is reset by writing a one to it.	0	RWC
26	DRES_C		Device status - reset change. This bit is set when the device received a bus reset. On a bus reset the device will automatically go to the default state (unconfigured and responding to address 0). The bit is reset by writing a one to it.	0	RWC
27	-		Reserved.	0	RO
28	VBUSDEBOUNCED		This bit indicates if Vbus is detected or not. The bit raises immediately when Vbus becomes high. It drops to zero if Vbus is low for at least 3 ms. If this bit is high and the DCon bit is set, the HW will enable the pull-up resistor to signal a connect.	0	RO
31:29	-		Reserved.	0	RO

15.6.2 USB Info register

Table 234. USB Info register (INFO, address 0x4008 0004) bit description

Bit	Symbol	Value	Description	Reset value	Access
10:0	FRAME_NR		Frame number. This contains the frame number of the last successfully received SOF. In case no SOF was received by the device at the beginning of a frame, the frame number returned is that of the last successfully received SOF. In case the SOF frame number contained a CRC error, the frame number returned will be the corrupted frame number as received by the device.	0	RO
14:11	ERR_CODE		The error code which last occurred:	0	RW
		0x0	No error		
		0x1	PID encoding error		
		0x2	PID unknown		
		0x3	Packet unexpected		
		0x4	Token CRC error		
		0x5	Data CRC error		
		0x6	Time out		
		0x7	Babble		
		0x8	Truncated EOP		
		0x9	Sent/Received NAK		
		0xA	Sent Stall		
		0xB	Overrun		
		0xC	Sent empty packet		
		0xD	Bitstuff error		
		0xE	Sync error		
		0xF	Wrong data toggle		
15	-		Reserved.	0	RO
31:16	-	-	Reserved	-	RO

15.6.3 USB EP Command/Status List start address

This 32-bit register indicates the start address of the USB EP Command/Status List.

Only a subset of these bits is programmable by software. The 8 least-significant bits are hardcoded to zero because the list must start on a 256 byte boundary. Bits 31 to 8 can be programmed by software.

Table 235. USB EP Command/Status List start address (EPLISTSTART, address 0x4008 0008) bit description

Bit	Symbol	Description	Reset value	Access
7:0	-	Reserved	0	RO
31:8	EP_LIST	Start address of the USB EP Command/Status List.	0	R/W

15.6.4 USB Data buffer start address

This register indicates the page of the AHB address where the endpoint data can be located.

Table 236. USB Data buffer start address (DATABUFSTART, address 0x4008 000C) bit description

Bit	Symbol	Description	Reset value	Access
21:0	-	Reserved	0	R
31:22	DA_BUF	Start address of the buffer pointer page where all endpoint data buffers are located.	0	R/W

15.6.5 USB Link Power Management register

Table 237. Link Power Management register (LPM, address 0x4008 0010) bit description

Bit	Symbol	Description	Reset value	Access
3:0	HIRD_HW	Host Initiated Resume Duration - HW. This is the HIRD value from the last received LPM token	0	RO
7:4	HIRD_SW	Host Initiated Resume Duration - SW. This is the time duration required by the USB device system to come out of LPM initiated suspend after receiving the host initiated LPM resume.	0	R/W
8	DATA_PENDING	As long as this bit is set to one and LPM supported bit is set to one, HW will return a NYET handshake on every LPM token it receives. If LPM supported bit is set to one and this bit is zero, HW will return an ACK handshake on every LPM token it receives. If SW has still data pending and LPM is supported, it must set this bit to 1.	0	R/W
31:9	RESERVED	Reserved	0	RO

15.6.6 USB Endpoint skip

Table 238. USB Endpoint skip (EPSKIP, address 0x4008 0014) bit description

Bit	Symbol	Description	Reset value	Access
29:0	SKIP	Endpoint skip: Writing 1 to one of these bits, will indicate to HW that it must deactivate the buffer assigned to this endpoint and return control back to software. When HW has deactivated the endpoint, it will clear this bit, but it will not modify the EPINUSE bit. An interrupt will be generated when the Active bit goes from 1 to 0. Note: In case of double-buffering, HW will only clear the Active bit of the buffer indicated by the EPINUSE bit.	0	R/W
31:30	-	Reserved	0	R

15.6.7 USB Endpoint Buffer in use

Table 239. USB Endpoint Buffer in use (EPINUSE, address 0x4008 0018) bit description

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved. Fixed to zero because the control endpoint zero is fixed to single-buffering for each physical endpoint.	0	R
9:2	BUF	Buffer in use: This register has one bit per physical endpoint. 0: HW is accessing buffer 0. 1: HW is accessing buffer 1.	0	R/W
31:10	-	Reserved	0	R

15.6.8 USB Endpoint Buffer Configuration

Table 240. USB Endpoint Buffer Configuration (EPBUFCFG, address 0x4008 001C) bit description

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved. Fixed to zero because the control endpoint zero is fixed to single-buffering for each physical endpoint.	0	R
9:2	BUF_SB	Buffer usage: This register has one bit per physical endpoint. 0: Single-buffer. 1: Double-buffer. If the bit is set to single-buffer (0), it will not toggle the corresponding EPINUSE bit when it clears the active bit. If the bit is set to double-buffer (1), HW will toggle the EPINUSE bit when it clears the Active bit for the buffer.	0	R/W
31:10	-	Reserved	0	R

15.6.9 USB interrupt status register

Table 241. USB interrupt status register (INTSTAT, address 0x4008 0020) bit description

Bit	Symbol	Description	Reset value	Access
0	EP0OUT	Interrupt status register bit for the Control EP0 OUT direction. This bit will be set if NBytes transitions to zero or the skip bit is set by software or a SETUP packet is successfully received for the control EP0. If the INTONNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP0 OUT direction. Software can clear this bit by writing a one to it.	0	R/WC
1	EP0IN	Interrupt status register bit for the Control EP0 IN direction. This bit will be set if NBytes transitions to zero or the skip bit is set by software. If the INTONNAK_CI is set, this bit will also be set when a NAK is transmitted for the Control EP0 IN direction. Software can clear this bit by writing a one to it.	0	R/WC
2	EP1OUT	Interrupt status register bit for the EP1 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the INTONNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP1 OUT direction. Software can clear this bit by writing a one to it.	0	R/WC
3	EP1IN	Interrupt status register bit for the EP1 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the INTONNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP1 IN direction. Software can clear this bit by writing a one to it.	0	R/WC
4	EP2OUT	Interrupt status register bit for the EP2 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the INTONNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP2 OUT direction. Software can clear this bit by writing a one to it.	0	R/WC
5	EP2IN	Interrupt status register bit for the EP2 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the INTONNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP2 IN direction. Software can clear this bit by writing a one to it.	0	R/WC
6	EP3OUT	Interrupt status register bit for the EP3 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the INTONNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP3 OUT direction. Software can clear this bit by writing a one to it.	0	R/WC

Table 241. USB interrupt status register (INTSTAT, address 0x4008 0020) bit description

Bit	Symbol	Description	Reset value	Access
7	EP3IN	Interrupt status register bit for the EP3 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the INTONNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP3 IN direction. Software can clear this bit by writing a one to it.	0	R/WC
8	EP4OUT	Interrupt status register bit for the EP4 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the INTONNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP4 OUT direction. Software can clear this bit by writing a one to it.	0	R/WC
9	EP4IN	Interrupt status register bit for the EP4 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the INTONNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP4 IN direction. Software can clear this bit by writing a one to it.	0	R/WC
29:10	-	Reserved	0	RO
30	FRAME_INT	Frame interrupt. This bit is set to one every millisecond when the VBUSDEBOUNCED bit and the DCON bit are set. This bit can be used by software when handling isochronous endpoints. Software can clear this bit by writing a one to it.	0	R/WC
31	DEV_INT	Device status interrupt. This bit is set by HW when one of the bits in the Device Status Change register are set. Software can clear this bit by writing a one to it.	0	R/WC

15.6.10 USB interrupt enable register

Table 242. USB interrupt enable register (INTEN, address 0x4008 0024) bit description

Bit	Symbol	Description	Reset value	Access
9:0	EP_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line indicated by the corresponding USB interrupt routing bit.	0	R/W
29:10	-	Reserved	0	RO
30	FRAME_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line indicated by the corresponding USB interrupt routing bit.	0	R/W
31	DEV_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line indicated by the corresponding USB interrupt routing bit.	0	R/W

15.6.11 USB set interrupt status register

Table 243. USB set interrupt status register (INTSETSTAT, address 0x4008 0028) bit description

Bit	Symbol	Description	Reset value	Access
9:0	EP_SET_INT	If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0	R/W
29:10	-	Reserved	0	RO
30	FRAME_SET_INT	If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0	R/W
31	DEV_SET_INT	If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0	R/W

15.6.12 USB interrupt routing register

Table 244. USB interrupt routing register (INTROUTING, address 0x4008 002C) bit description

Bit	Symbol	Description	Reset value	Access
9:0	ROUTE_INT9_0	This bit can control on which hardware interrupt line the interrupt will be generated: 0: IRQ interrupt line is selected for this interrupt bit 1: FIQ interrupt line is selected for this interrupt bit	0	R/W
29:10	-	Reserved	0	RO
30	ROUTE_INT30	This bit can control on which hardware interrupt line the interrupt will be generated: 0: IRQ interrupt line is selected for this interrupt bit 1: FIQ interrupt line is selected for this interrupt bit	0	R/W
31	ROUTE_INT31	This bit can control on which hardware interrupt line the interrupt will be generated: 0: IRQ interrupt line is selected for this interrupt bit 1: FIQ interrupt line is selected for this interrupt bit	0	R/W

15.6.13 USB Endpoint toggle

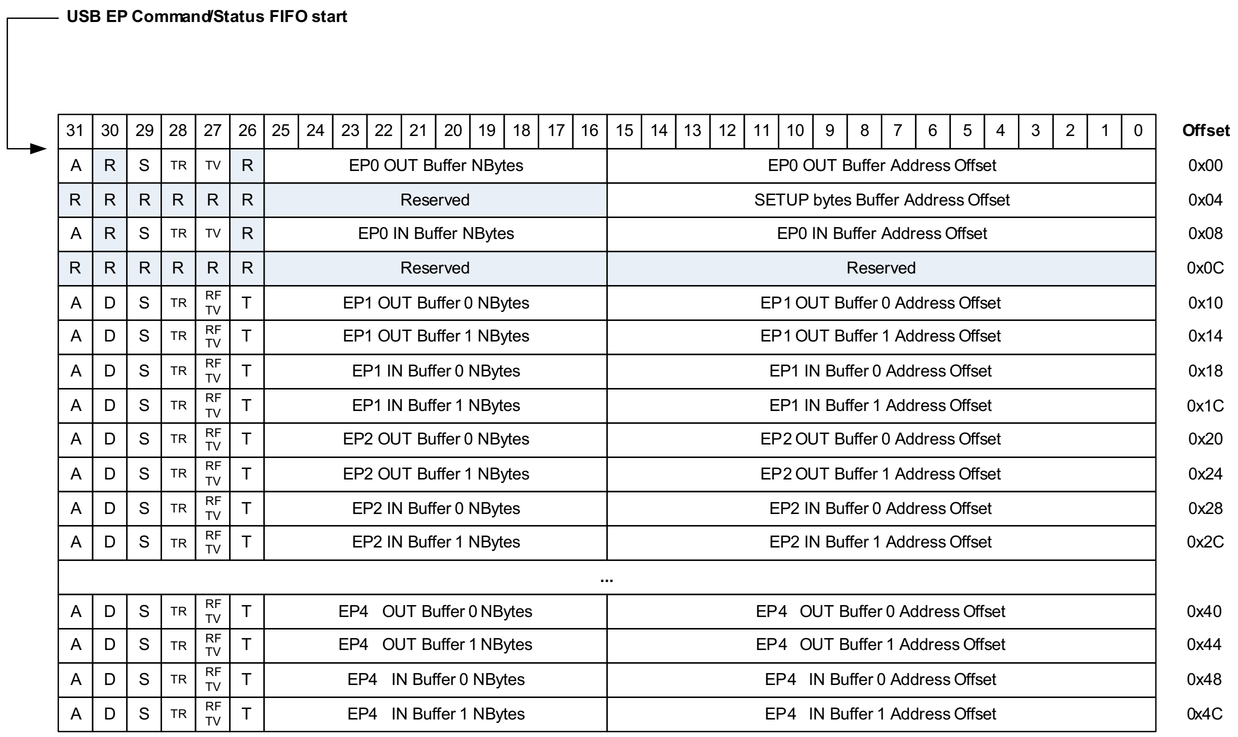
Table 245. USB Endpoint toggle (EPTOGGLE, address 0x4008 0034) bit description

Bit	Symbol	Description	Reset value	Access
9:0	TOGGLE	Endpoint data toggle: This field indicates the current value of the data toggle for the corresponding endpoint.	0	R
31:10	-	Reserved	0	R

15.7 Functional description

15.7.1 Endpoint command/status list

Figure 57 gives an overview on how the Endpoint List is organized in memory. The USB EP Command/Status List start register points to the start of the list that contains all the endpoint information in memory. The order of the endpoints is fixed as shown in the picture.



31	30	29	28	27	26	25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	**Offset**
A	R	S	TR	TV	R	EP0 OUT Buffer NBytes	EP0 OUT Buffer Address Offset	0x00
R	R	R	R	R	R	Reserved	SETUP bytes Buffer Address Offset	0x04
A	R	S	TR	TV	R	EP0 IN Buffer NBytes	EP0 IN Buffer Address Offset	0x08
R	R	R	R	R	R	Reserved	Reserved	0x0C
A	D	S	TR	RF TV	T	EP1 OUT Buffer 0 NBytes	EP1 OUT Buffer 0 Address Offset	0x10
A	D	S	TR	RF TV	T	EP1 OUT Buffer 1 NBytes	EP1 OUT Buffer 1 Address Offset	0x14
A	D	S	TR	RF TV	T	EP1 IN Buffer 0 NBytes	EP1 IN Buffer 0 Address Offset	0x18
A	D	S	TR	RF TV	T	EP1 IN Buffer 1 NBytes	EP1 IN Buffer 1 Address Offset	0x1C
A	D	S	TR	RF TV	T	EP2 OUT Buffer 0 NBytes	EP2 OUT Buffer 0 Address Offset	0x20
A	D	S	TR	RF TV	T	EP2 OUT Buffer 1 NBytes	EP2 OUT Buffer 1 Address Offset	0x24
A	D	S	TR	RF TV	T	EP2 IN Buffer 0 NBytes	EP2 IN Buffer 0 Address Offset	0x28
A	D	S	TR	RF TV	T	EP2 IN Buffer 1 NBytes	EP2 IN Buffer 1 Address Offset	0x2C
...								
A	D	S	TR	RF TV	T	EP4 OUT Buffer 0 NBytes	EP4 OUT Buffer 0 Address Offset	0x40
A	D	S	TR	RF TV	T	EP4 OUT Buffer 1 NBytes	EP4 OUT Buffer 1 Address Offset	0x44
A	D	S	TR	RF TV	T	EP4 IN Buffer 0 NBytes	EP4 IN Buffer 0 Address Offset	0x48
A	D	S	TR	RF TV	T	EP4 IN Buffer 1 NBytes	EP4 IN Buffer 1 Address Offset	0x4C

Fig 57. Endpoint command/status list (see also Table 246)

Table 246. Endpoint commands

Symbol	Access	Description
A	RW	Active The buffer is enabled. HW can use the buffer to store received OUT data or to transmit data on the IN endpoint. Software can only set this bit to '1'. As long as this bit is set to one, software is not allowed to update any of the values in this 32-bit word. In case software wants to deactivate the buffer, it must write a one to the corresponding "skip" bit in the USB Endpoint skip register. Hardware can only write this bit to zero. It will do this when it receives a short packet or when the NBytes field transitions to zero or when software has written a one to the "skip" bit.
D	RW	Disabled 0: The selected endpoint is enabled. 1: The selected endpoint is disabled. If a USB token is received for an endpoint that has the disabled bit set, hardware will ignore the token and not return any data or handshake. When a bus reset is received, software must set the disable bit of all endpoints to 1. Software can only modify this bit when the active bit is zero.
S	RW	Stall 0: The selected endpoint is not stalled 1: The selected endpoint is stalled The Active bit has always higher priority than the Stall bit. This means that a Stall handshake is only sent when the active bit is zero and the stall bit is one. Software can only modify this bit when the active bit is zero.
TR	RW	Toggle Reset When software sets this bit to one, the HW will set the toggle value equal to the value indicated in the "toggle value" (TV) bit. For the control endpoint zero, this is not needed to be used because the hardware resets the endpoint toggle to one for both directions when a setup token is received. For the other endpoints, the toggle can only be reset to zero when the endpoint is reset.
RF / TV	RW	Rate Feedback mode / Toggle value For bulk endpoints and isochronous endpoints this bit is reserved and must be set to zero. For the control endpoint zero this bit is used as the toggle value. When the toggle reset bit is set, the data toggle is updated with the value programmed in this bit. When the endpoint is used as an interrupt endpoint, it can be set to the following values. 0: Interrupt endpoint in 'toggle mode' 1: Interrupt endpoint in 'rate feedback mode'. This means that the data toggle is fixed to zero for all data packets. When the interrupt endpoint is in 'rate feedback mode', the TR bit must always be set to zero.

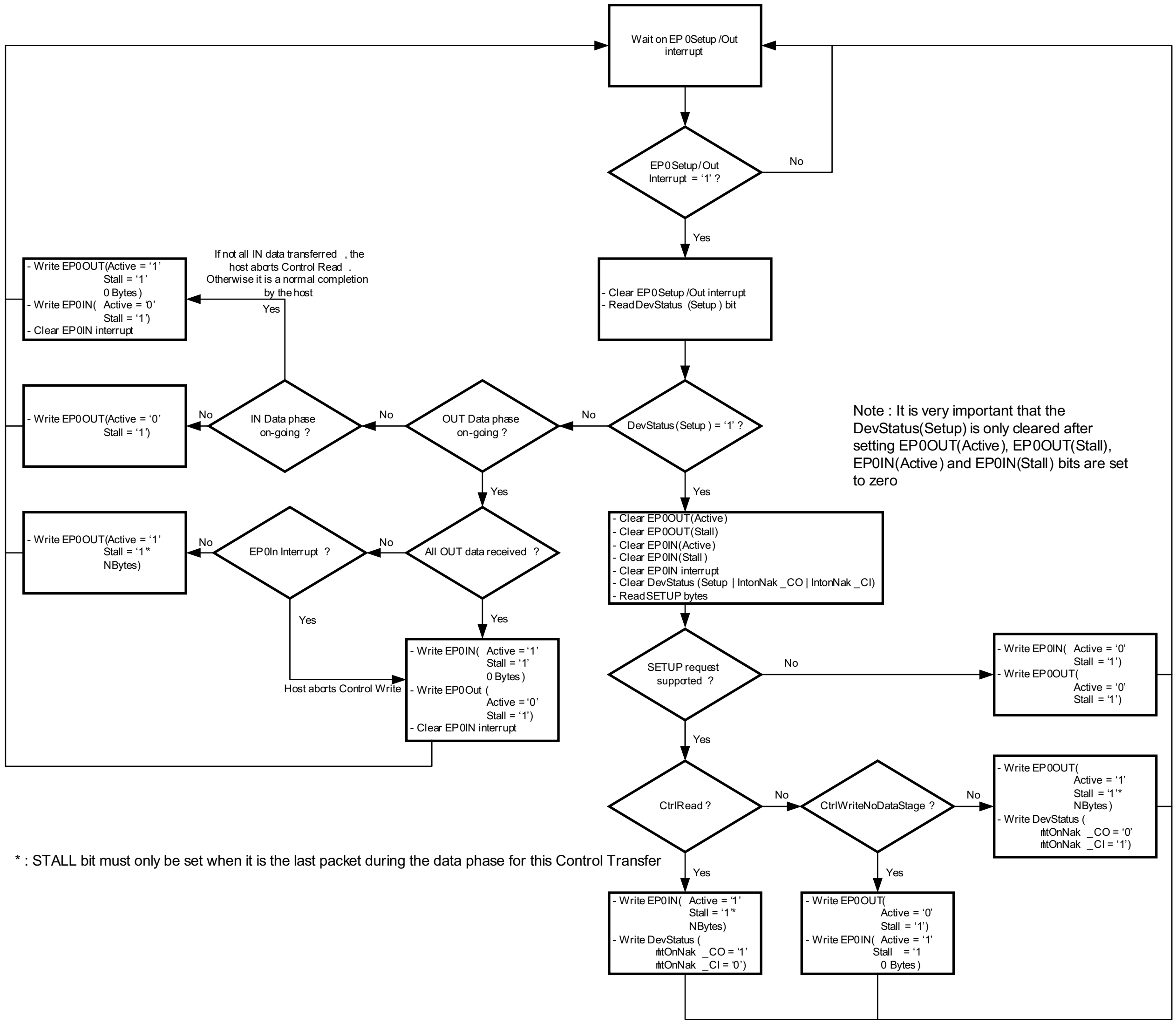
Table 246. Endpoint commands

Symbol	Access	Description
T	RW	Endpoint Type 0: Generic endpoint. The endpoint is configured as a bulk or interrupt endpoint 1: Isochronous endpoint
NBytes	RW	For OUT endpoints this is the number of bytes that can be received in this buffer. For IN endpoints this is the number of bytes that must be transmitted. HW decrements this value with the packet size every time when a packet is successfully transferred. Note: If a short packet is received on an OUT endpoint, the active bit will be cleared and the NBytes value indicates the remaining buffer space that is not used. Software calculates the received number of bytes by subtracting the remaining NBytes from the programmed value.
Address Offset	RW	Bits 21 to 6 of the buffer start address. The address offset is updated by hardware after each successful reception/transmission of a packet. Hardware increments the original value with the integer value when the packet size is divided by 64. Examples: • If an isochronous packet of 200 bytes is successfully received, the address offset is incremented by 3. • If a packet of 64 bytes is successfully received, the address offset is incremented by 1. • If a packet of less than 64 bytes is received, the address offset is not incremented.

Remark: When receiving a SETUP token for endpoint zero, the HW will only read the SETUP bytes Buffer Address offset to know where it has to store the received SETUP bytes. The hardware will ignore all other fields. In case the SETUP stage contains more than 8 bytes, it will only write the first 8 bytes to memory. A USB compliant host must never send more than 8 bytes during the SETUP stage.

For EP0 transfers, the hardware will do auto handshake as long as the ACTIVE bit is set in EP0_IN/OUT command list. Unlike other endpoints, the hardware will not clear the ACTIVE bit after transfer is done. Thus, the software should manually clear the bit whenever it receives new setup packet and set it only after it has queued the data for control transfer. See Figure 58 “Flowchart of control endpoint 0 - OUT direction”.

15.7.2 Control endpoint 0



Note : It is very important that the DevStatus(Setup) is only cleared after setting EP0OUT(Active), EP0OUT(Stall), EP0IN(Active) and EP0IN(Stall) bits are set to zero

* : STALL bit must only be set when it is the last packet during the data phase for this Control Transfer

Fig 58. Flowchart of control endpoint 0 - OUT direction

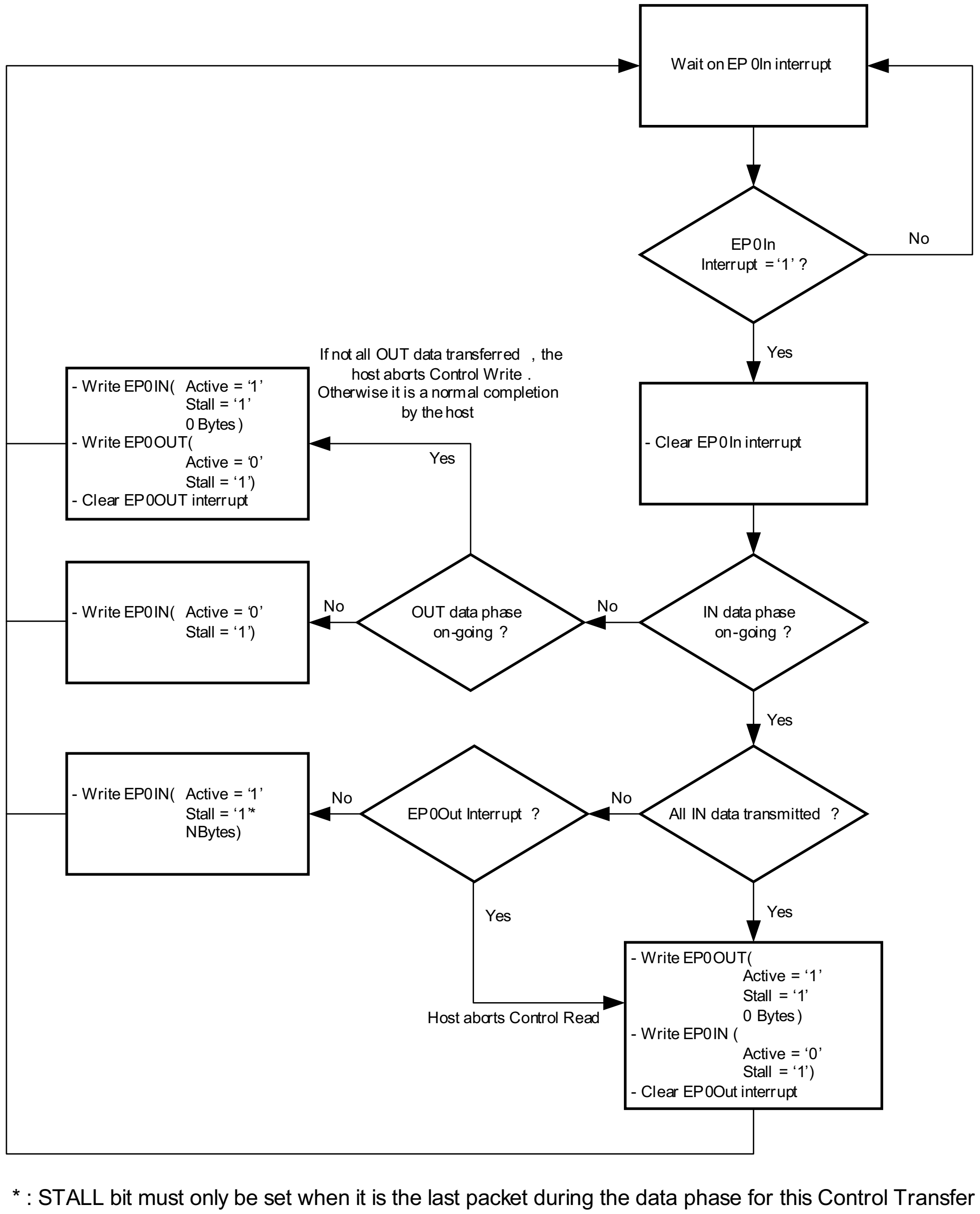


Fig 59. Flowchart of control endpoint 0 - IN direction

15.7.3 Generic endpoint: single-buffering

To enable single-buffering, software must set the corresponding "USB EP Buffer Config" bit to zero. In the "USB EP Buffer in use" register, software can indicate which buffer is used in this case.

When software wants to transfer data, it programs the different bits in the Endpoint command/status entry and sets the active bits. The hardware will transmit/receive multiple packets for this endpoint until the NBytes value is equal to zero. When NBytes goes to zero, hardware clears the active bit and sets the corresponding interrupt status bit.

Software must wait until hardware has cleared the Active bit to change some of the command/status bits. This prevents hardware from overwriting a new value programmed by software with some old values that were still cached.

If software wants to disable the active bit before the hardware has finished handling the complete buffer, it can do this by setting the corresponding endpoint skip bit in USB endpoint skip register.

15.7.4 Generic endpoint: double-buffering

To enable double-buffering, software must set the corresponding "USB EP Buffer Config" bit to one. The "USB EP Buffer in use" register indicates which buffer will be used by HW when the next token is received.

When HW clears the active bit of the current buffer in use, it will switch the buffer in use. Software can also force HW to use a certain buffer by writing to the "USB EP Buffer in use" bit.

15.7.5 Special cases

15.7.5.1 Use of the Active bit

The use of the Active bit is a bit different between OUT and IN endpoints.

When data must be received for the OUT endpoint, the software will set the Active bit to one and program the NBytes field to the maximum number of bytes it can receive.

When data must be transmitted for an IN endpoint, the software sets the Active bit to one and programs the NBytes field to the number of bytes that must be transmitted.

15.7.5.2 Generation of a STALL handshake

Special care must be taken when programming the endpoint to send a STALL handshake. A STALL handshake is only sent in the following situations:

- The endpoint is enabled (Disabled bit = 0)
- The active bit of the endpoint is set to 0. (No packet needs to be received/transmitted for that endpoint).
- The stall bit of the endpoint is set to one.

15.7.5.3 Clear Feature (endpoint halt)

When a non-control endpoint has returned a STALL handshake, the host will send a Clear Feature (Endpoint Halt) for that endpoint. When the device receives this request, the endpoint must be unstalled and the toggle bit for that endpoint must be reset back to zero. In order to do that the software must program the following items for the endpoint that is indicated.

If the endpoint is used in single-buffer mode, program the following:

- Set STALL bit (S) to 0.
- Set toggle reset bit (TR) to 1 and set toggle value bit (TV) to 0.

If the endpoint is used in double-buffer mode, program the following:

- Set the STALL bit of both buffer 0 and buffer 1 to 0.
- Read the buffer in use bit for this endpoint.
- Set the toggle reset bit (TR) to 1 and set the toggle value bit (TV) to 0 for the buffer indicated by the buffer in use bit.

15.7.5.4 Set configuration

When a SetConfiguration request is received with a configuration value different from zero, the device software must enable all endpoints that will be used in this configuration and reset all the toggle values. To do so, it must generate the procedure explained in Section 15.7.5.3 for every endpoint that will be used in this configuration.
For all endpoints that are not used in this configuration, it must set the Disabled bit (D) to one.

UM10732

Chapter 16: LPC11U6x/E6x 12-bit Analog-to-Digital Converter (ADC)

Rev. 1.8 — 19 December 2016 **User manual**

16.1 How to read this chapter

The ADC is available on all parts. The number of ADC channels depends on the package.

Table 247. ADC channels available

Package	ADC channels
LQFP48	ADC_1 to ADC_3, ADC_6 to ADC_9, ADC_11
LQFP64	ADC_0 to ADC_3, ADC_6 to ADC_11
LQFP100	ADC_0 to ADC_11

16.2 Features

- 12-bit successive approximation analog to digital converter.
- Input multiplexing among 12 pins and one internal source.
- Two configurable conversion sequences with independent triggers.
- Optional automatic high/low threshold comparison and “zero crossing” detection.
- Power-down mode and low-power operating mode.
- Measurement range VREFN to VREFP (typically 3 V; not to exceed VDDA voltage level).
- Maximum 12-bit conversion rate of 2 Msamples/s (V_{DDA} = 2.7 V to 3.6 V) or 1 Msamples/s (V_{DDA} = 2.4 V to 2.7 V).
- Burst conversion mode for single or multiple inputs.
- DMA support.

16.3 Basic configuration

Configure the ADC as follows:

- Use the SYSAHBCLKCTRL register (Table 40) to enable the clock to the ADC register interface and the ADC clock.
- The ADC block creates four interrupts. The ADC threshold crossing and end-of-sequence A interrupts are combined and connected to the ADC_A_IRQ (slot #24). The end-of-sequence B and overrun interrupts are combined and connected to the ADC_B_IRQ (slot # 29).
- The ADC analog inputs are selected in the IOCON block.See Table 83.
- The power to the ADC block is controlled by the PDRUNCFG register in the SYSCON block. See Table 69. Once the ADC is powered by the PDRUNCFG register bit, the low-power mode bit in the ADC CTRL register can be used to turn off the ADC when it is not sampling and turn on the ADC automatically when any of the ADC conversion triggers are raised. See Table 251 and Section 16.7.5.

- Calibration is required after every power-up or wake-up from Deep power-down mode. See Section 16.3.4 “Hardware self-calibration”.
- The temperature sensor output is connected to ADC channel 0 whenever the temperature sensor is powered in the PDRUNCFG register (Table 69). If the temperature sensor is powered down (default), then channel 0 is connected to pin ADC_0 by default.
- The maximum sampling rate depends on V_{DDA}. To obtain the maximum sampling rate, set the ADC clock and the ADC clock divider CLKDIV for either 50 MHz (2 Msamples/s; VDDA >= 2.7 V) or 25 MHz (1 Msamples/s; VDDA < 2.7 V). See Table 251.
- Configure the ADC for the appropriate analog supply voltage using the TRM register (Table 264). The default setting assumes $V_{DDA} \geq 2.7$ V.

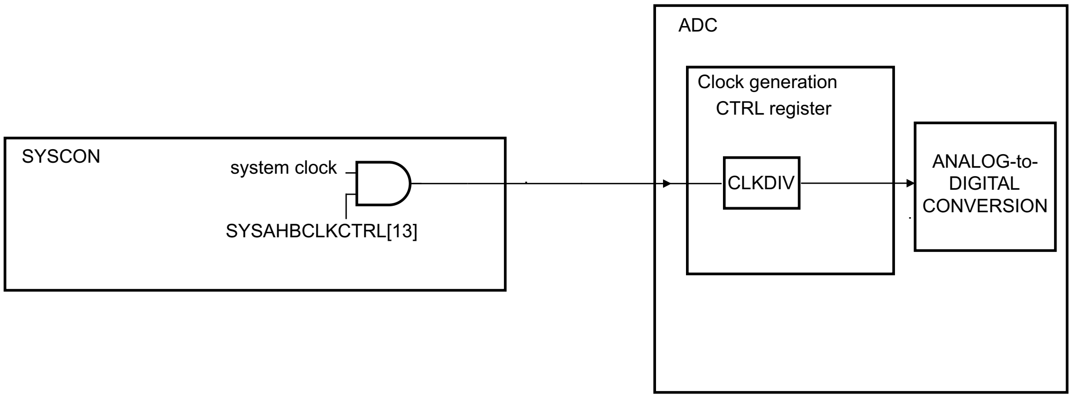


Fig 60. ADC clocking

16.3.1 Perform a single ADC conversion using a software trigger

Remark: When A/D conversions are triggered by software only and hardware triggers are not used in the conversion sequence, follow these steps to avoid spurious conversions:

1. Before changing the trigger set-up, disable the conversion sequence by setting the SEQ_ENA bit to 0 in the SEQA_CTRL register.
2. Ensure that the signal connected to the hardware trigger through the TRIGGER bits is not toggling and LOW. See Table 248. This is the default for the TXEV signal.
3. Set the TRIGPOL bit to 1 in the in the SEQA_CTRL register.

Once the sequence is enabled again, the ADC converts a sample whenever the START bit is written to. The TRIGPOL bit can be set in the same write that sets the SEQ_ENA and the START bits. Be careful not to modify the TRIGGER, TRIGPOL, and SEQ_ENA bits on subsequent writes to the START bit. See also Section 16.7.2.1 “Avoiding spurious hardware triggers”.

The ADC converts an analog input signal VIN on the ADC_[11:0]. The VREFP and VREFN pins provide a positive and negative reference voltage input. The result of the conversion is (4095 x VIN)/(VREFP - VREFN). The result of an input voltage below VREFN is 0, and the result of an input voltage above VREFP is 4095 (0xFFF).

To perform a single ADC conversion for ADC0 channel 1 using the analog signal on pin ADC_1, follow these steps:

1. Enable the analog function ADC_1.
2. Configure the system clock to be 50 MHz and select a CLKDIV value of 0 for a sampling rate of 2 Msamples/s using the ADC CTRL register.
3. Select ADC channel 1 to perform the conversion by setting the CHANNELS bits to 0x2 in the SEQA_CTL register.
4. Set the TRIGPOL bit to 1 and the SEQA_ENA bit to 1 in the SEQA_CTRL register.
5. Set the START bit to 1 in the SEQA_CTRL register.
6. Read the RESULT bits in the DAT1 register for the conversion result.

16.3.2 Perform a sequence of conversions triggered by an external pin

The ADC can perform conversions on a sequence of selected channels. Each individual conversion of the sequence (single-step) or the entire sequence can be triggered by hardware. Hardware triggers are either a signal from an external pin or an internal signal. See Section 16.3.3.

To perform a single-step conversion on the first four channels of ADC0 triggered by a rising edge on CT16B0_CAP0 pin, follow these steps:

1. Enable the analog function ADC_0 to ADC_3.See Table 83.
2. Configure the system clock to be 50 MHz and select a CLKDIV value of 0 for a sampling rate of 2 Msamples/s using the ADC CTRL register.
3. Select ADC channels 0 to 3 to perform the conversion by setting the CHANNELS bits to 0xF in the SEQA_CTL register.
4. Select CT16B0_CAP0 by writing 0x5to the TRIGGER bits in the SEQA_CTRL register.
5. To generate one interrupt at the end of the entire sequence, set the MODE bit to 1 in the SEQA_CTRL register.
6. Select single-step mode by setting the SINGLESTEP bit in the SEQA_CTRL register to 1.
7. Enable the Sequence A by setting the SEQA_ENA bit.

   A conversion on ADC0 channel 0 will be triggered whenever the pin PIO1_0 goes from LOW to HIGH. The conversion on the next channel (channel 1) is triggered on the next rising edge of CT16B0_CAP0. The ADC_A interrupt is generated when the sequence has finished after four rising edges on CT16B0_CAP0.
8. Read the RESULT bits in the DAT0 to DAT3 registers for the conversion result.

16.3.3 ADC hardware trigger inputs

An analog-to-digital conversion can be initiated by a hardware trigger. You can select the trigger independently for each of the two conversion sequences in the ADC SEQA_CTRL or SEQB_CTRL registers by programming the hardware trigger input # into the TRIGGER bits.

Related registers:

- Table 252 "A/D Conversion Sequence A Control Register (SEQA_CTRL, address 0x4001 C008) bit description"
- Table 253 "A/D Conversion Sequence B Control Register (SEQB_CTRL, address 0x4001 C00C) bit description"

Table 248. ADC hardware trigger inputs

Input #	Source	Description
0	ARM_TXEV	ARM Cortex M0+ generated event
1	CT32B0_MAT0	Match output 0 of 32-bit timer CT32B0
2	CT32B0_MAT1 or SCT0_OUT0	Match output 1 of CT32B0 ORed with output 0 of SCT0.
3	CT16B0_MAT0	Match output 0 of 16-bit timer CT16B0
4	CT16B0_MAT1 or SCT1_OUT0	Match output 1 of CT16B0 ORed with output 0 of SCT1.
5	CT16B0_CAP0	Capture input 0 of 16-bit timer CT16B0
6	CT16B1_CAP0	Capture input 0 of 16-bit timer CT16B1
7	CT32B0_CAP0	Capture input 0 of 32-bit timer CT32B0

16.3.4 Hardware self-calibration

The A/D converter includes a built-in, hardware self-calibration mode. In order to achieve the specified ADC accuracy, the A/D converter must be recalibrated following every chip reset before initiating normal ADC operation.

The calibration voltage level is VREFP - VREFN.

To calibrate the ADC follow these steps:

1. Save the current contents of the ADC CTRL register if different from default.
2. In a single write to the ADC CTRL register, do the following to start the calibration:
   - Set the calibration mode bit CALMODE.
   - Write a divider value to the CLKDIV bit field that divides the system clock to yield an ADC clock of about 500 kHz.
   - Clear the LPWR bit.
3. Poll the CALMODE bit until it is cleared.

Before launching a new A/D conversion, restore the contents of the CTRL register or use the default values.

A calibration cycle requires approximately 290 μs to complete. While calibration is in progress, normal ADC conversions cannot be launched, and the ADC Control Register must not be written to. The calibration procedure does not use the CPU or memory, so other processes can be executed during calibration.

16.4 Pin description

The ADC cell can measure the voltage on any of the input signals on the analog input channel. Digital signals are disconnected from the ADC input pins when the ADC function is selected on that pin in the IOCON register.

Remark: If the ADC is used, signal levels on analog input pins must not be above the level of V_{DDA} at any time. Otherwise, ADC readings will be invalid. If the ADC is not used in an application, then the pins associated with ADC inputs can be configured as digital I/O pins and are 5 V tolerant.

The VREFP and VREFN pins provide a positive and negative reference voltage input. The result of the conversion is (4095 x input voltage VIN)/(VREFP - VREFN). The result of an input voltage below VREFN is 0, and the result of an input voltage above VREFP is 4095 (0xFFF).

When the ADC is not used, tie V_{DDA} and VREFP to VDD and V_{SSA} and VREFP to V_{SS}.

Analog Power and Ground should typically be the same voltages as V_{DD} and V_{SS}, but should be isolated to minimize noise and error.

Remark: For best performance, select VREFP and VREFN at the same voltage levels as V_{DDA} and V_{SSA}. When selecting VREFP and VREFN different from V_{DDA} and V_{SSA}, ensure that the voltage midpoints are the same:

$$(VREFP-VREFN)/2 + VREFN = V_{DDA}/2$$

Table 249. ADC pin description

Function	Direction	Description
V_{REFP}	Ref	Positive voltage reference. VREFP must be > 2.4 V. For best performance, select VREFP = V_{DDA} and VREFN = V_{SSA}.
V_{REFN}	Ref	Negative voltage reference
V_{DDA}	Supply	ADC power supply
V_{SSA}	Supply	ADC ground
ADC_0	AI	Analog input channel 0.
ADC_1	AI	Analog input channel 1.
ADC_2	AI	Analog input channel 2.
ADC_3	AI	Analog input channel 3.
ADC_4	AI	Analog input channel 4.
ADC_5	AI	Analog input channel 5.
ADC_6	AI	Analog input channel 6.
ADC_7	AI	Analog input channel 7.
ADC_8	AI	Analog input channel 8.

Table 249. ADC pin description

Function	Direction	Description
ADC_9	AI	Analog input channel 9.
ADC_10	AI	Analog input channel 10.
ADC_11	AI	Analog input channel 11.

16.4.1 ADC vs. digital receiver

The ADC function must be selected via the IOCON registers in order to get accurate voltage readings on the monitored pin. The MODE bits in the IOCON register should also disable both pull-up and pull-down resistors. For a pin hosting an ADC input, it is not possible to have a have a digital function selected and yet get valid ADC readings. An inside circuit disconnects ADC hardware from the associated pin whenever a digital function is selected on that pin.

16.5 General description

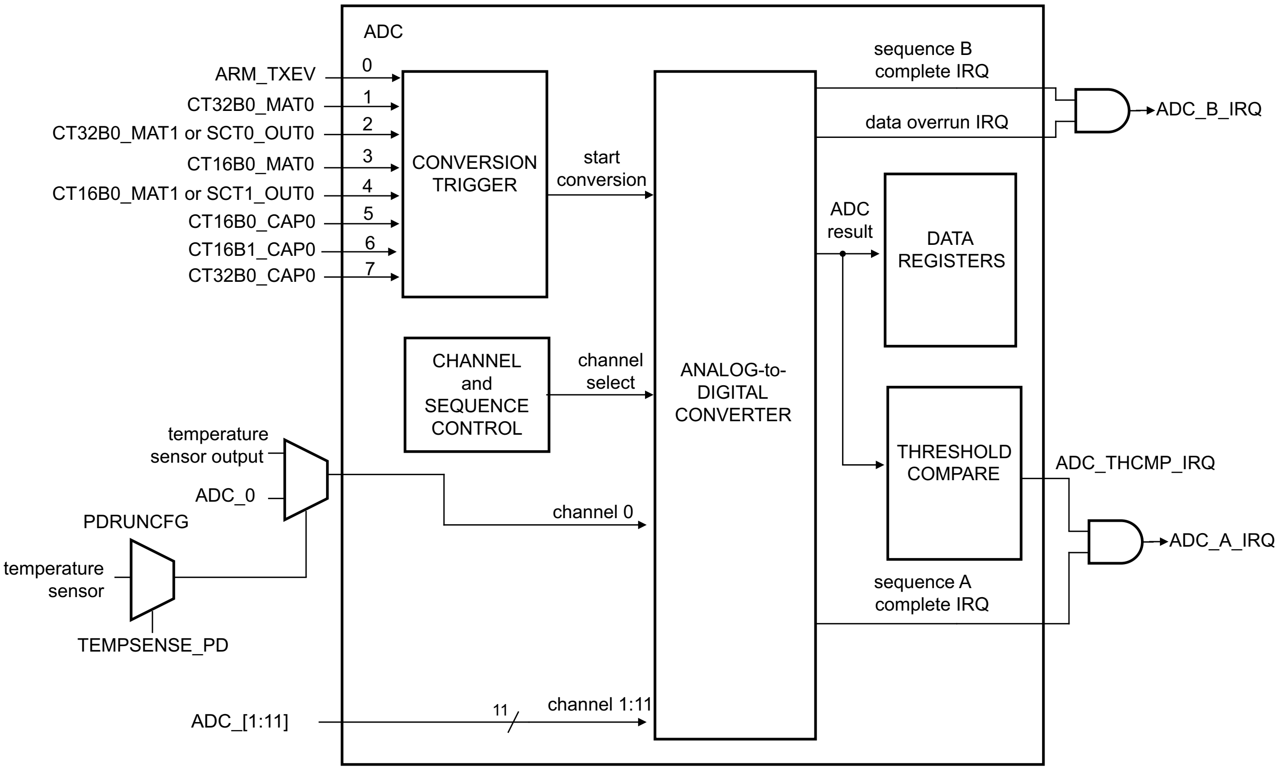


Fig 61. ADC block diagram

The ADC controller provides great flexibility in launching and controlling sequences of A/D conversions using the associated 12-bit, successive approximation A/D converter. A/D conversion sequences can be initiated under software control or in response to a selected hardware trigger. The ADC supports eight hardware triggers.

Once the triggers are set up (software and hardware triggers can be mixed), the ADC runs through the pre-defined conversion sequence, converting a sample whenever a trigger signal arrives, until the sequence is disabled.

The ADC controller uses the system clock as a bus clock. The ADC clock is derived from the system clock. A programmable divider is included to scale the system clock to the maximum ADC clock rate of 50 MHz. The ADC clock drives the successive approximation process.

A fully accurate conversion requires 25 of these ADC clocks.

16.6 Register description

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

Table 250. Register overview : ADC (base address 0x4001 C000)

Name	Access	Address offset	Description	Reset value	Reference
CTRL	R/W	0x000	A/D Control Register. Contains the clock divide value, enable bits for each sequence and the A/D power-down bit.	0x0	Table 251
-	-	0x004	Reserved.	-	-
SEQA_CTRL	R/W	0x008	A/D Conversion Sequence-A control Register: Controls triggering and channel selection for conversion sequence-A. Also specifies interrupt mode for sequence-A.	0x0	Table 252
SEQB_CTRL	R/W	0x00C	A/D Conversion Sequence-B Control Register: Controls triggering and channel selection for conversion sequence-B. Also specifies interrupt mode for sequence-B.	0x0	Table 253
SEQA_GDAT	R/W	0x010	A/D Sequence-A Global Data Register. This register contains the result of the most recent A/D conversion performed under sequence-A	NA	Table 254
SEQB_GDAT	R/W	0x014	A/D Sequence-B Global Data Register. This register contains the result of the most recent A/D conversion performed under sequence-B	NA	Table 255
DAT0	RO	0x020	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0.	NA	Table 256
DAT1	RO	0x024	A/D Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1.	NA	Table 256
DAT2	RO	0x028	A/D Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2.	NA	Table 256
DAT3	RO	0x02C	A/D Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3.	NA	Table 256
DAT4	RO	0x030	A/D Channel 4 Data Register. This register contains the result of the most recent conversion completed on channel 4.	NA	Table 256
DAT5	RO	0x034	A/D Channel 5 Data Register. This register contains the result of the most recent conversion completed on channel 5.	NA	Table 256
DAT6	RO	0x038	A/D Channel 6 Data Register. This register contains the result of the most recent conversion completed on channel 6.	NA	Table 256
DAT7	RO	0x03C	A/D Channel 7 Data Register. This register contains the result of the most recent conversion completed on channel 7.	NA	Table 256

Table 250. Register overview : ADC (base address 0x4001 C000)

Name	Access	Address offset	Description	Reset value	Reference
DAT8	RO	0x040	A/D Channel 8 Data Register. This register contains the result of the most recent conversion completed on channel 7.	NA	Table 256
DAT9	RO	0x044	A/D Channel 9 Data Register. This register contains the result of the most recent conversion completed on channel 7.	NA	Table 256
DAT10	RO	0x048	A/D Channel 10 Data Register. This register contains the result of the most recent conversion completed on channel 7.	NA	Table 256
DAT11	RO	0x04C	A/D Channel 11 Data Register. This register contains the result of the most recent conversion completed on channel 7.	NA	Table 256
THR0_LOW	R/W	0x050	A/D Low Compare Threshold Register 0 : Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 0.	0x0	Table 257
THR1_LOW	R/W	0x054	A/D Low Compare Threshold Register 1: Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 1.	0x0	Table 258
THR0_HIGH	R/W	0x058	A/D High Compare Threshold Register 0: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 0.	0x0	Table 259
THR1_HIGH	R/W	0x05C	A/D High Compare Threshold Register 1: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 1.	0x0	Table 260
CHAN_THRSEL	R/W	0x060	A/D Channel-Threshold Select Register. Specifies which set of threshold compare registers are to be used for each channel	0x0	Table 261
INTEN	R/W	0x064	A/D Interrupt Enable Register. This register contains enable bits that enable the sequence-A, sequence-B, threshold compare and data overrun interrupts to be generated.	0x0	Table 262
FLAGS	R/W	0x068	A/D Flags Register. Contains the four interrupt request flags and the individual component overrun and threshold-compare flags. (The overrun bits replicate information stored in the result registers).	0x0	Table 263
TRM	R/W	0x06C	ADC trim register.	0x0000 0F00	Table 264

16.6.1 ADC Control Register

This register specifies the clock divider value to be used to generate the ADC clock and general operating mode bits including a low power mode that allows the A/D to be turned off to save power when not in use.

Table 251. A/D Control Register (CTRL, addresses 0x4001 C000) bit description

Bit	Symbol	Value	Description	Reset value
7:0	CLKDIV		The system clock is divided by this value plus one to produce the sampling clock. The sampling clock should be less than or equal to 50 MHz (for 2 Msamples/s). Typically, software should program the smallest value in this field that yields this maximum clock rate or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable.	0
9:8	-		Reserved.Do not write a one to these bits.	0
10	LPWRMODE		Select low-power ADC mode. The analog circuitry is automatically powered-down when no conversions are taking place. When any (hardware or software) triggering event is detected, the analog circuitry is enabled. After the required start-up time, the requested conversion will be launched. Once the conversion completes, the analog-circuitry will again be powered-down provided no further conversions are pending. Using this mode can save an appreciable amount of current (approximately 2.5 mA) when conversions are required relatively infrequently. The penalty for using this mode is an approximately 15 ADC clock delay, based on the frequency specified in the CLKDIV field, from the time the trigger event occurs until sampling of the A/D input commences. **Remark:** This mode will NOT power-up the ADC when the ADC analog block is powered down in the system control block.	0
		0	Disabled. The low-power ADC mode is disabled. The analog circuitry remains activated even when no conversions are requested.	
		1	Enabled. The low-power ADC mode is enabled.	
29:11			Reserved, do not write ones to reserved bits.	0
30	CALMODE		Writing a 1 to this bit initiates a self-calibration cycle. This bit will be automatically cleared by hardware after the calibration cycle is complete. **Remark:** Other bits of this register may be written to concurrently with setting this bit, however once this bit has been set no further writes to this register are permitted until the full calibration cycle has ended.	0
31	-		Reserved.	0

16.6.2 A/D Conversion Sequence A Control Register

There are two, independent conversion sequences that can be configured, each consisting of a set of conversions on one or more channels. This control register specifies the channel selection and trigger conditions for the A sequence and contains bits to allow software to initiate that conversion sequence.

To avoid conversions on spurious triggers, only change the trigger configuration when the conversion sequence is disabled. A conversion can be triggered by software or hardware in the conversion sequence, but if conversions are triggered by software only, spurious hardware triggers must be prevented. See Section 16.3.1 “Perform a single ADC conversion using a software trigger”.

Remark: Set the BURST and SEQU_ENA bits at the same time.

Table 252: A/D Conversion Sequence A Control Register (SEQA_CTRL, address 0x4001 C008) bit description

Bit	Symbol	Value	Description	Reset value
11:0	CHANNELS		Selects which one or more of the twelve channels will be sampled and converted when this sequence is launched. A 1 in any bit of this field will cause the corresponding channel to be included in the conversion sequence, where bit 0 corresponds to channel 0, bit 1 to channel 1 and so forth. When this conversion sequence is triggered, either by a hardware trigger or via software command, A/D conversions will be performed on each enabled channel, in sequence, beginning with the lowest-ordered channel. **Remark:** This field can ONLY be changed while the SEQA_ENA bit (bit 31) is LOW. It is allowed to change this field and set bit 31 in the same write.	0x00
14:12	TRIGGER		Selects which of the available hardware trigger sources will cause this conversion sequence to be initiated. Program the trigger input number in this field. **Remark:** In order to avoid generating a spurious trigger, it is recommended writing to this field only when the SEQA_ENA bit (bit 31) is low. It is safe to change this field and set bit 31 in the same write.	0x0
17:15	-		Reserved.	-
18	TRIGPOL		Select the polarity of the selected input trigger for this conversion sequence. **Remark:** In order to avoid generating a spurious trigger, it is recommended writing to this field only when the SEQA_ENA bit (bit 31) is low. It is safe to change this field and set bit 31 in the same write.	0
		0	Negative edge. A negative edge launches the conversion sequence on the selected trigger input.	
		1	Positive edge. A positive edge launches the conversion sequence on the selected trigger input.	
19	SYNCBYPASS		Setting this bit allows the hardware trigger input to bypass synchronization flip-flops stages and therefore shorten the time between the trigger input signal and the start of a conversion. There are slightly different criteria for whether or not this bit can be set depending on the clock operating mode: Synchronous mode: Synchronization may be bypassed (this bit may be set) if the selected trigger source is already synchronous with the main system clock (eg. coming from an on-chip, system-clock-based timer). Whether this bit is set or not, a trigger pulse must be maintained for at least one system clock period. Asynchronous mode: Synchronization may be bypassed (this bit may be set) if it is certain that the duration of a trigger input pulse will be at least one cycle of the ADC clock (regardless of whether the trigger comes from and on-chip or off-chip source). If this bit is NOT set, the trigger pulse must at least be maintained for one system clock period.	0
		0	Enable synchronization. The hardware trigger bypass is not enabled.	
		1	Bypass synchronization. The hardware trigger bypass is enabled.	
25:20	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	N/A
26	START		Writing a 1 to this field will launch one pass through this conversion sequence. The behavior will be identical to a sequence triggered by a hardware trigger. Do not write 1 to this bit if the BURST bit is set. **Remark:** This bit is only set to a 1 momentarily when written to launch a conversion sequence. It will consequently always read-back as a zero.	0

Table 252: A/D Conversion Sequence A Control Register (SEQA_CTRL, address 0x4001 C008) bit description

Bit	Symbol	Value	Description	Reset value
27	BURST		Writing a 1 to this bit will cause this conversion sequence to be continuously cycled through. Other sequence A triggers will be ignored while this bit is set. Repeated conversions can be halted by clearing this bit. The sequence currently in progress will be completed before conversions are terminated.	0
28	SINGLESTEP		When this bit is set, a hardware trigger or a write to the START bit will launch a single conversion on the next channel in the sequence instead of the default response of launching an entire sequence of conversions. Once all of the channels comprising a sequence have been converted, a subsequent trigger will repeat the sequence beginning with the first enabled channel. Interrupt generation will still occur either after each individual conversion or at the end of the entire sequence, depending on the state of the MODE bit.	0
29	LOWPRIO		Set priority for sequence A.	0
		0	Low priority. Any B trigger which occurs while an A conversion sequence is active will be ignored and lost.	
		1	High priority. Setting this bit to a 1 will permit any enabled B sequence trigger (including a B sequence software start) to immediately interrupt this sequence and launch a B sequence in it’s place. The conversion currently in progress will be terminated. The A sequence that was interrupted will automatically resume after the B sequence completes. The channel whose conversion was terminated will be re-sampled and the conversion sequence will resume from that point.	
30	MODE		Indicates whether the primary method for retrieving conversion results for this sequence will be accomplished via reading the global data register (SEQA_GDAT) at the end of each conversion, or the individual channel result registers at the end of the entire sequence. Impacts when conversion-complete interrupt/DMA triggers for sequence-A will be generated and which overrun conditions contribute to an overrun interrupt as described below:	0
		0	End of conversion. The sequence A interrupt/DMA flag will be set at the end of each individual A/D conversion performed under sequence A. This flag will mirror the DATAVALID bit in the SEQA_GDAT register. The OVERRUN bit in the SEQA_GDAT register will contribute to generation of an overrun interrupt if enabled.	
		1	End of sequence. The sequence A interrupt/DMA flag will be set when the entire set of sequence-A conversions completes. This flag will need to be explicitly cleared by software or by the DMA-clear signal in this mode. The OVERRUN bit in the SEQA_GDAT register will NOT contribute to generation of an overrun interrupt/DMA trigger since it is assumed this register may not be utilized in this mode.	

Table 252: A/D Conversion Sequence A Control Register (SEQA_CTRL, address 0x4001 C008) bit description

Bit	Symbol	Value	Description	Reset value
31	SEQA_ENA		Sequence Enable. In order to avoid spuriously triggering the sequence, care should be taken to only set the SEQA_ENA bit when the selected trigger input is in its INACTIVE state (as defined by the TRIGPOL bit). If this condition is not met, the sequence will be triggered immediately upon being enabled.	0
		0	Disabled. Sequence A is disabled. Sequence A triggers are ignored. If this bit is cleared while sequence A is in progress, the sequence will be halted at the end of the current conversion. After the sequence is re-enabled, a new trigger will be required to restart the sequence beginning with the next enabled channel.	
		1	Enabled. Sequence A is enabled.	

16.6.3 A/D Conversion Sequence B Control Register

There are two, independent conversion sequences that can be configured, each consisting of a set of conversions on one or more channels. This control register specifies the channel selection and trigger conditions for the B sequence, as well bits to allow software to initiate that conversion sequence.

To avoid conversions on spurious triggers, only change the trigger configuration when the conversion sequence is disabled. A conversion can be triggered by software or hardware in the conversion sequence, but if conversions are triggered by software only, spurious hardware triggers must be prevented. See Section 16.3.1 “Perform a single ADC conversion using a software trigger”.

Remark: Set the BURST and SEQU_ENA bits at the same time.

Table 253: A/D Conversion Sequence B Control Register (SEQB_CTRL, address 0x4001 C00C) bit description

Bit	Symbol	Value	Description	Reset value
11:0	CHANNELS		Selects which one or more of the twelve channels will be sampled and converted when this sequence is launched. A 1 in any bit of this field will cause the corresponding channel to be included in the conversion sequence, where bit 0 corresponds to channel 0, bit 1 to channel 1 and so forth. When this conversion sequence is triggered, either by a hardware trigger or via software command, A/D conversions will be performed on each enabled channel, in sequence, beginning with the lowest-ordered channel. **Remark:** This field can ONLY be changed while the SEQB_ENA bit (bit 31) is LOW. It is permissible to change this field and set bit 31 in the same write.	0x00
14:12	TRIGGER		Selects which of the available hardware trigger sources will cause this conversion sequence to be initiated. Program the trigger input number in this field. **Remark:** In order to avoid generating a spurious trigger, it is recommended writing to this field only when the SEQA_ENA bit (bit 31) is low. It is safe to change this field and set bit 31 in the same write.	0x0
17:15	-		Reserved.	-
18	TRIGPOL		Select the polarity of the selected input trigger for this conversion sequence. **Remark:** In order to avoid generating a spurious trigger, it is recommended writing to this field only when the SEQA_ENA bit (bit 31) is low. It is safe to change this field and set bit 31 in the same write.	0
		0	Negative edge. A negative edge launches the conversion sequence on the selected trigger input.	
		1	Positive edge. A positive edge launches the conversion sequence on the selected trigger input.	
19	SYNCBYPASS		Setting this bit allows the hardware trigger input to bypass synchronization flip-flops stages and therefore shorten the time between the trigger input signal and the start of a conversion. There are slightly different criteria for whether or not this bit can be set depending on the clock operating mode: Synchronous mode: Synchronization may be bypassed (this bit may be set) if the selected trigger source is already synchronous with the main system clock (eg. coming from an on-chip, system-clock-based timer). Whether this bit is set or not, a trigger pulse must be maintained for at least one system clock period. Asynchronous mode: Synchronization may be bypassed (this bit may be set) if it is certain that the duration of a trigger input pulse will be at least one cycle of the ADC clock (regardless of whether the trigger comes from and on-chip or off-chip source). If this bit is NOT set, the trigger pulse must at least be maintained for one system clock period.	0
		0	Enable synchronization. The hardware trigger bypass is not enabled.	
		1	Bypass synchronization. The hardware trigger bypass is enabled.	

Table 253: A/D Conversion Sequence B Control Register (SEQB_CTRL, address 0x4001 C00C) bit description

Bit	Symbol	Value	Description	Reset value
25:20	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	N/A
26	START		Writing a 1 to this field will launch one pass through this conversion sequence. The behavior will be identical to a sequence triggered by a hardware trigger. Do not write a 1 to this bit if the BURST bit is set. **Remark:** This bit is only set to a 1 momentarily when written to launch a conversion sequence. It will consequently always read-back as a zero.	0
27	BURST		Writing a 1 to this bit will cause this conversion sequence to be continuously cycled through. Other B triggers will be ignored while this bit is set. Repeated conversions can be halted by clearing this bit. The sequence currently in progress will be completed before conversions are terminated.	0
28	SINGLESTEP		When this bit is set, a hardware trigger or a write to the START bit will launch a single conversion on the next channel in the sequence instead of the default response of launching an entire sequence of conversions. Once all of the channels comprising a sequence have been converted, a subsequent trigger will repeat the sequence beginning with the first enabled channel. Interrupt generation will still occur either after each individual conversion or at the end of the entire sequence, depending on the state of the MODE bit.	0
29	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	N/A
30	MODE		Indicates whether the primary method for retrieving conversion results for this sequence will be accomplished via reading the global data register (SEQB_GDAT) at the end of each conversion, or the individual channel result registers at the end of the entire sequence. Impacts when conversion-complete interrupt/DMA trigger for sequence-B will be generated and which overrun conditions contribute to an overrun interrupt as described below:	0
		0	End of conversion. The sequence B interrupt/DMA flag will be set at the end of each individual A/D conversion performed under sequence B. This flag will mirror the DATAVALID bit in the SEQB_GDAT register. The OVERRUN bit in the SEQB_GDAT register will contribute to generation of an overrun interrupt if enabled.	
		1	End of sequence. The sequence B interrupt/DMA flag will be set when the entire set of sequence B conversions completes. This flag will need to be explicitly cleared by software or by the DMA-clear signal in this mode. The OVERRUN bit in the SEQB_GDAT register will NOT contribute to generation of an overrun interrupt since it is assumed this register will not be utilized in this mode.	

Table 253: A/D Conversion Sequence B Control Register (SEQB_CTRL, address 0x4001 C00C) bit description

Bit	Symbol	Value	Description	Reset value
31	SEQB_ENA		Sequence Enable. In order to avoid spuriously triggering the sequence, care should be taken to only set the SEQA_ENA bit when the selected trigger input is in its INACTIVE state (as defined by the TRIGPOL bit). If this condition is not met, the sequence will be triggered immediately upon being enabled.	0
		0	Disabled. Sequence B is disabled. Sequence B triggers are ignored. If this bit is cleared while sequence B is in progress, the sequence will be halted at the end of the current conversion. After the sequence is re-enabled, a new trigger will be required to restart the sequence beginning with the next enabled channel.	
		1	Enabled. Sequence B is enabled.	

16.6.4 A/D Global Data Register A and B

The A/D Global Data Registers contain the result of the most recent A/D conversion completed under each conversion sequence.

Results of A/D conversions can be read in one of two ways. One is to use these A/D Global Data Registers to read data from the ADC at the end of each A/D conversion. Another is to read the individual A/D Channel Data Registers, typically after the entire sequence has completed. It is recommended to use one method consistently for a given conversion sequence.

The global registers are useful in conjunction with DMA operation - particularly when the channels selected for conversion are not sequential (hence the addresses of the individual result registers will not be sequential, making it difficult for the DMA engine to address them). For interrupt-driven code it will more likely be advantageous to wait for an entire sequence to complete and then retrieve the results from the individual channel registers.

Remark: The method to be employed for each sequence should be reflected in the MODE bit in the corresponding ADSEQn_CTRL register since this will impact interrupt and overrun flag generation.

Table 254: A/D Sequence A Global Data Register (SEQA_GDAT, address 0x4001 C010) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:4	RESULT	This field contains the 12-bit A/D conversion result from the most recent conversion performed under conversion sequence associated with this register. The result is the a binary fraction representing the voltage on the currently-selected input channel as it falls within the range of V_{REFP} to V_{REFN}. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on V_{REFN}, while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on V_{REFP}. DATAVALID = 1 indicates that this result has not yet been read.	NA
17:16	THCMPRANGE	Indicates whether the result of the last conversion performed was above, below or within the range established by the designated threshold comparison registers (THRn_LOW and THRn_HIGH).	
19:18	THCMPCROSS	Indicates whether the result of the last conversion performed represented a crossing of the threshold level established by the designated LOW threshold comparison register (THRn_LOW) and, if so, in what direction the crossing occurred.	
25:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
29:26	CHN	These bits contain the channel from which the RESULT bits were converted (e.g. 0000 identifies channel 0, 0001 channel 1...).	NA
30	OVERRUN	This bit is set if a new conversion result is loaded into the RESULT field before a previous result has been read - i.e. while the DATAVALID bit is set. This bit is cleared, along with the DATAVALID bit, whenever this register is read. This bit will contribute to an overrun interrupt request if the MODE bit (in SEQA_CTRL) for the corresponding sequence is set to ‘0’ (and if the overrun interrupt is enabled).	0
31	DATAVALID	This bit is set to ‘1’ at the end of each conversion when a new result is loaded into the RESULT field. It is cleared whenever this register is read. This bit will cause a conversion-complete interrupt for the corresponding sequence if the MODE bit (in SEQA_CTRL) for that sequence is set to 0 (and if the interrupt is enabled).	0

Table 255: A/D Sequence B Global Data Register (SEQB_GDAT, address 0x4001 C014) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:4	RESULT	This field contains the 12-bit A/D conversion result from the most recent conversion performed under conversion sequence associated with this register. This will be a binary fraction representing the voltage on the currently-selected input channel as it falls within the range of V_{REFP} to V_{REFN}. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on V_{REFN}, while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on V_{REFP}. DATAVALID = 1 indicates that this result has not yet been read.	NA
17:16	THCMPRANGE	Indicates whether the result of the last conversion performed was above, below or within the range established by the designated threshold comparison registers (THRn_LOW and THRn_HIGH). Threshold Range Comparison result. 0x0 = In Range: The last completed conversion was greater than or equal to the value programmed into the designated LOW threshold register (THRn_LOW) but less than or equal to the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x1 = Below Range: The last completed conversion on was less than the value programmed into the designated LOW threshold register (THRn_LOW). 0x2 = Above Range: The last completed conversion was greater than the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x3 = Reserved.	
19:18	THCMPCROSS	Indicates whether the result of the last conversion performed represented a crossing of the threshold level established by the designated LOW threshold comparison register (THRn_LOW) and, if so, in what direction the crossing occurred. 0x0 = No threshold Crossing detected: The most recent completed conversion on this channel had the same relationship (above or below) to the threshold value established by the designated LOW threshold register (THRn_LOW) as did the previous conversion on this channel. 0x1 = Reserved. 0x2 = Downward Threshold Crossing Detected. Indicates that a threshold crossing in the downward direction has occurred - i.e. the previous sample on this channel was above the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is below that threshold. 0x3 = Upward Threshold Crossing Detected. Indicates that a threshold crossing in the upward direction has occurred - i.e. the previous sample on this channel was below the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is above that threshold.	
25:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 255: A/D Sequence B Global Data Register (SEQB_GDAT, address 0x4001 C014) bit description

Bit	Symbol	Description	Reset value
29:26	CHN	These bits contain the channel from which the RESULT bits were converted (e.g. 0b0000 identifies channel 0, 0b0001 channel 1...).	NA
30	OVERRUN	This bit is set if a new conversion result is loaded into the RESULT field before a previous result has been read - i.e. while the DATAVALID bit is set. This bit is cleared, along with the DATAVALID bit, whenever this register is read. This bit will contribute to an overrun interrupt request if the MODE bit (in SEQB_CTRL) for the corresponding sequence is set to 0 (and if the overrun interrupt is enabled).	0
31	DATAVALID	This bit is set to 1 at the end of each conversion when a new result is loaded into the RESULT field. It is cleared whenever this register is read. This bit will cause a conversion-complete interrupt for the corresponding sequence if the MODE bit (in SEQB_CTRL) for that sequence is set to 0 (and if the interrupt is enabled).	0

16.6.5 A/D Channel Data Registers 0 to 11

The A/D Channel Data Registers hold the result of the last conversion completed for each A/D channel. They also include status bits to indicate when a conversion has been completed, when a data overrun has occurred, and where the most recent conversion fits relative to the range dictated by the high and low threshold registers.

Results of A/D conversion can be read in one of two ways. One is to use the A/D Global Data Registers for each of the sequences to read data from the ADC at the end of each A/D conversion. Another is to use these individual A/D Channel Data Registers, typically after the entire sequence has completed. It is recommended to use one method consistently for a given conversion sequence.

Remark: The method to be employed for each sequence should be reflected in the MODE bit in the corresponding SEQ_CTRL register since this will impact interrupt and overrun flag generation.

The information presented in the DAT registers always pertains to the most recent conversion completed on that channel regardless of what sequence requested the conversion or which trigger caused it.

The OVERRUN fields for each channel are also replicated in the FLAGS register.

Table 256. A/D Data Registers (DAT[0:11], address 0x4001 C020 (DAT0) to 0x4001 C04C (DAT11)) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:4	RESULT	This field contains the 12-bit A/D conversion result from the last conversion performed on this channel. This will be a binary fraction representing the voltage on the AD0[n] pin, as it falls within the range of V_{REFP} to V_{REFN}. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on V_{REFN}, while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on V_{REFP}.	NA
17:16	THCMPRANGE	Threshold Range Comparison result. 0x0 = In Range: The last completed conversion was greater than or equal to the value programmed into the designated LOW threshold register (THRn_LOW) but less than or equal to the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x1 = Below Range: The last completed conversion on was less than the value programmed into the designated LOW threshold register (THRn_LOW). 0x2 = Above Range: The last completed conversion was greater than the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x3 = Reserved.	NA
19:18	THCMPCROSS	Threshold Crossing Comparison result. 0x0 = No threshold Crossing detected: The most recent completed conversion on this channel had the same relationship (above or below) to the threshold value established by the designated LOW threshold register (THRn_LOW) as did the previous conversion on this channel. 0x1 = Reserved. 0x2 = Downward Threshold Crossing Detected. Indicates that a threshold crossing in the downward direction has occurred - i.e. the previous sample on this channel was above the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is below that threshold. 0x3 = Upward Threshold Crossing Detected. Indicates that a threshold crossing in the upward direction has occurred - i.e. the previous sample on this channel was below the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is above that threshold.	NA
25:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 256. A/D Data Registers (DAT[0:11], address 0x4001 C020 (DAT0) to 0x4001 C04C (DAT11)) bit description

Bit	Symbol	Description	Reset value
29:26	CHANNEL	This field is hard-coded to contain the channel number that this particular register relates to (i.e. this field will contain 0b0000 for the DAT0 register, 0b0001 for the DAT1 register, etc)	NA
30	OVERRUN	This bit will be set to a 1 if a new conversion on this channel completes and overwrites the previous contents of the RESULT field before it has been read - i.e. while the DONE bit is set. This bit is cleared, along with the DONE bit, whenever this register is read or when the data related to this channel is read from either of the global SEQn_GDAT registers. This bit (in any of the 12 registers) will cause an overrun interrupt request to be asserted if the overrun interrupt is enabled. **Remark:** While it is allowed to include the same channels in both conversion sequences, doing so may cause erratic behavior of the DONE and OVERRUN bits in the data registers associated with any of the channels that are shared between the two sequences. Any erratic OVERRUN behavior will also affect overrun interrupt generation, if enabled.	NA
31	DATAVALID	This bit is set to 1 when an A/D conversion on this channel completes. This bit is cleared whenever this register is read or when the data related to this channel is read from either of the global SEQn_GDAT registers. **Remark:** While it is allowed to include the same channels in both conversion sequences, doing so may cause erratic behavior of the DONE and OVERRUN bits in the data registers associated with any of the channels that are shared between the two sequences. Any erratic OVERRUN behavior will also affect overrun interrupt generation, if enabled.	NA

16.6.6 A/D Compare Low Threshold Registers 0 and 1

These registers set the LOW threshold levels against which A/D conversions on all channels will be compared.

Each channel will either be compared to the THR0_LOW/HIGH registers or to the THR1_LOW/HIGH registers depending on what is specified for that channel in the CHAN_THRSEL register.

A conversion result LESS THAN this value on any channel will cause the THCMP_RANGE status bits for that channel to be set to 0b01. This result will also generate an interrupt request if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

If, for two successive conversions on a given channel, one result is below this threshold and the other is equal-to or above this threshold, than a threshold crossing has occurred. In this case the MSB of the THCMP_CROSS status bits will indicate that a threshold crossing has occurred and the LSB will indicate the direction of the crossing. A threshold crossing event will also generate an interrupt request if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

Table 257. A/D Compare Low Threshold register 0 (THR0_LOW, address 0x4001 C050) bit description

Bit	Symbol	Description	Reset value
3:0		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:4	THRLOW	Low threshold value against which A/D results will be compared	0x000
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 258. A/D Compare Low Threshold register 1 (THR1_LOW, address 0x4001 C054) bit description

Bit	Symbol	Description	Reset value
3:0		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:4	THRLOW	Low threshold value against which A/D results will be compared	0x000
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

16.6.7 A/D Compare High Threshold Registers 0 and 1

These registers set the HIGH threshold level against which A/D conversions on all channels will be compared.

Each channel will either be compared to the THR0_LOW/HIGH registers or to the THR1_LOW/HIGH registers depending on what is specified for that channel in the CHAN_THRSEL register.

A conversion result greater than this value on any channel will cause the THCMP status bits for that channel to be set to 0b10. This result will also generate an interrupt request if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

Table 259: Compare High Threshold register0 (THR0_HIGH, address 0x4001 C058) bit description

Bit	Symbol	Description	Reset value
3:0		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:4	THRHIGH	High threshold value against which A/D results will be compared	0x000
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 260: Compare High Threshold register 1 (THR1_HIGH, address 0x4001 C05C) bit description

Bit	Symbol	Description	Reset value
3:0		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:4	THRHIGH	High threshold value against which A/D results will be compared	0x000
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

16.6.8 A/D Channel Threshold Select register

For each channel, this register indicates which pair of threshold registers conversion results should be compared to.

Table 261: A/D Channel Threshold Select register (CHAN_THRSEL, addresses 0x4001 C060) bit description

Bit	Symbol	Value	Description	Reset value
0	CH0_THRSEL		Threshold select by channel.	0
		0	Threshold 0. Channel 0 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers	
		1	Threshold 1. Channel 0 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers	
1	CH1_THRSEL		Threshold select by channel.	0
		0	Threshold 0. Channel 1 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers	
		1	Threshold 1. Channel 1 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers	
2	CH2_THRSEL		Threshold select by channel.	0
		0	Threshold 0. Channel 2 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers	
		1	Threshold 1. Channel 2 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers	
3	CH3_THRSEL		Threshold select by channel.	0
		0	Threshold 0. Channel 3 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers	
		1	Threshold 1. Channel 3 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers	
4	CH4_THRSEL		Threshold select by channel.	0
		0	Threshold 0. Channel 4 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers	
		1	Threshold 1. Channel 4 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers	
5	CH5_THRSEL		Threshold select by channel.	0
		0	Threshold 0. Channel 5 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers	
		1	Threshold 1. Channel 5 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers	

Table 261: A/D Channel Threshold Select register (CHAN_THRSEL, addresses 0x4001 C060) bit description

Bit	Symbol	Value	Description	Reset value
6	CH6_THRSEL		Threshold select by channel.	0
		0	Threshold 0. Channel 6 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers	
		1	Threshold 1. Channel 6 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers	
7	CH7_THRSEL		Threshold select by channel.	0
		0	Threshold 0. Channel 7 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers	
		1	Threshold 1. Channel 7 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers	
8	CH8_THRSEL		Threshold select by channel.	0
		0	Threshold 0. Channel 8 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers	
		1	Threshold 1. Channel 8 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers	
9	CH9_THRSEL		Threshold select by channel.	0
		0	Threshold 0. Channel 9 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers	
		1	Threshold 1. Channel 9 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers	
10	CH10_THRSEL		Threshold select by channel.	0
		0	Threshold 0. Channel 10 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers	
		1	Threshold 1. Channel 10 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers	
11	CH11_THRSEL		Threshold select by channel.	0
		0	Threshold 0. Channel 11 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers	
		1	Threshold 1. Channel 11 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers	
31:12			Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

16.6.9 A/D Interrupt Enable Register

There are four separate interrupt requests generated by the ADC: conversion-complete or sequence-complete interrupts for each of the two sequences, a threshold-comparison out-of-range interrupt, and a data overrun interrupt. The two conversion/sequence-complete interrupts can also serve as DMA triggers.

These interrupts may be combined into one request on some chips if there is a limited number of interrupt slots. This register contains the interrupt-enable bits for each interrupt.

In this register, threshold events selected in the ADCMPINTENn bits are described as follows:

- Disabled: Threshold comparisons on channel n will not generate an A/D threshold-compare interrupt request.
- Outside threshold: A conversion result on channel n which is outside the range specified by the designated HIGH and LOW threshold registers will set the channel n THCMP flag in the FLAGS register and generate an A/D threshold-compare interrupt request.
- Crossing threshold: Detection of a threshold crossing on channel n will set the channel n THCMP flag in the ADFLAGS register and generate an A/D threshold-compare interrupt request.

Remark: Overrun and threshold-compare interrupts related to a particular channel will occur regardless of which sequence was in progress at the time the conversion was performed or what trigger caused the conversion.

Table 262: A/D Interrupt Enable register (INTEN, address 0x4001 C064) bit description

Bit	Symbol	Value	Description	Reset value
0	SEQA_INTEN		Sequence A interrupt enable.	0
		0	Disabled. The sequence A interrupt/DMA trigger is disabled.	
		1	Enabled. The sequence A interrupt/DMA trigger is enabled and will be asserted either upon completion of each individual conversion performed as part of sequence A, or upon completion of the entire A sequence of conversions, depending on the MODE bit in the SEQA_CTRL register.	
1	SEQB_INTEN		Sequence B interrupt enable.	0
		0	Disabled. The sequence B interrupt/DMA trigger is disabled.	
		1	Enabled. The sequence B interrupt/DMA trigger is enabled and will be asserted either upon completion of each individual conversion performed as part of sequence B, or upon completion of the entire B sequence of conversions, depending on the MODE bit in the SEQB_CTRL register.	
2	OVR_INTEN		Overrun interrupt enable.	0
		0	Disabled. The overrun interrupt is disabled.	
		1	Enabled. The overrun interrupt is enabled. Detection of an overrun condition on any of the 12 channel data registers will cause an overrun interrupt request. In addition, if the MODE bit for a particular sequence is 0, then an overrun in the global data register for that sequence will also cause this interrupt request to be asserted.	
4:3	ADCMPINTEN0		Threshold comparison interrupt enable.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		0x2	Crossing threshold.	
		0x3	Reserved	
6:5	ADCMPINTEN1		Threshold comparison interrupt enable.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		0x2	Crossing threshold.	
		0x3	Reserved.	

Table 262: A/D Interrupt Enable register (INTEN, address 0x4001 C064) bit description

Bit	Symbol	Value	Description	Reset value
8:7	ADCMPINTEN2		Threshold comparison interrupt enable.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		0x2	Crossing threshold.	
		0x3	Reserved	
10:9	ADCMPINTEN3		Threshold comparison interrupt enable.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		0x2	Crossing threshold.	
		0x3	Reserved	
12:11	ADCMPINTEN4		Threshold comparison interrupt enable.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		0x2	Crossing threshold.	
		0x3	Reserved	
14:13	ADCMPINTEN5		Threshold comparison interrupt enable.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		0x2	Crossing threshold.	
		0x3	Reserved	
16:15	ADCMPINTEN6		Threshold comparison interrupt enable.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		0x2	Crossing threshold.	
		0x3	Reserved.	
18:17	ADCMPINTEN7		Threshold comparison interrupt enable.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		0x2	Crossing threshold.	
		0x3	Reserved	
20:19	ADCMPINTEN8		Threshold comparison interrupt enable.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		0x2	Crossing threshold.	
		0x3	Reserved	
22:21	ADCMPINTEN9		Threshold comparison interrupt enable.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		0x2	Crossing threshold.	
		0x3	Reserved	

Table 262: A/D Interrupt Enable register (INTEN, address 0x4001 C064) bit description

Bit	Symbol	Value	Description	Reset value
24:23	ADCMPINTEN10		Threshold comparison interrupt enable.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		0x2	Crossing threshold.	
		0x3	Reserved	
26:25	ADCMPINTEN11		Threshold comparison interrupt enable.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		0x2	Crossing threshold.	
		0x3	Reserved	
31:27	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

16.6.10 A/D Flag register

The A/D Flags registers contains the four interrupt request flags along with the individual overrun flags that contribute to an overrun interrupt and the component threshold-comparison flags that contribute to that interrupt.

The channel OVERRUN flags, mirror those in the appearing in the individual ADDAT registers for each channel, indicate a data overrun in each of those registers.

Likewise, the SEQA_OVR and SEQB_OVR bits mirror the OVERRUN bits in the two global data registers (SEQA_GDAT and SEQB_GDAT).

Remark: The SEQn_INT conversion/sequence-complete flags also serve as DMA triggers.

Table 263: A/D Flags register (FLAGS, address 0x4001 C068) bit description

Bit	Symbol	Description	Reset value
0	THCMP0	Threshold comparison event on Channel 0. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.	0
1	THCMP1	Threshold comparison event on Channel 1. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.	0
2	THCMP2	Threshold comparison event on Channel 2. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.	0
3	THCMP3	Threshold comparison event on Channel 3. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.	0
4	THCMP4	Threshold comparison event on Channel 4. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.	0

Table 263: A/D Flags register (FLAGS, address 0x4001 C068) bit description

Bit	Symbol	Description	Reset value
5	THCMP5	Threshold comparison event on Channel 5. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.	0
6	THCMP6	Threshold comparison event on Channel 6. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.	0
7	THCMP7	Threshold comparison event on Channel 7. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.	0
8	THCMP8	Threshold comparison event on Channel 8. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.	0
9	THCMP9	Threshold comparison event on Channel 9. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.	0
10	THCMP10	Threshold comparison event on Channel 10. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.	0
11	THCMP11	Threshold comparison event on Channel 11. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.	0
12	OVERRUN0	Mirrors the OVERRRUN status flag from the result register for A/D channel 0	0
13	OVERRUN1	Mirrors the OVERRRUN status flag from the result register for A/D channel 1	0
14	OVERRUN2	Mirrors the OVERRRUN status flag from the result register for A/D channel 2	0
15	OVERRUN3	Mirrors the OVERRRUN status flag from the result register for A/D channel 3	0
16	OVERRUN4	Mirrors the OVERRRUN status flag from the result register for A/D channel 4	0
17	OVERRUN5	Mirrors the OVERRRUN status flag from the result register for A/D channel 5	0
18	OVERRUN6	Mirrors the OVERRRUN status flag from the result register for A/D channel 6	0
19	OVERRUN7	Mirrors the OVERRRUN status flag from the result register for A/D channel 7	0
20	OVERRUN8	Mirrors the OVERRRUN status flag from the result register for A/D channel 8	0
21	OVERRUN9	Mirrors the OVERRRUN status flag from the result register for A/D channel 9	0
22	OVERRUN10	Mirrors the OVERRRUN status flag from the result register for A/D channel 10	0
23	OVERRUN11	Mirrors the OVERRRUN status flag from the result register for A/D channel 11	0
24	SEQA_OVR	Mirrors the global OVERRUN status flag in the SEQA_GDAT register	0
25	SEQB_OVR	Mirrors the global OVERRUN status flag in the SEQB_GDAT register	0
27:26	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
28	SEQA_INT	Sequence A interrupt/DMA flag. If the MODE bit in the SEQA_CTRL register is 0, this flag will mirror the DATAVALID bit in the sequence A global data register (SEQA_GDAT), which is set at the end of every A/D conversion performed as part of sequence A. It will be cleared automatically when the SEQA_GDAT register is read. If the MODE bit in the SEQA_CTRL register is 1, this flag will be set upon completion of an entire A sequence. In this case it must be cleared by writing a 1 to this SEQA_INT bit. This interrupt must be enabled in the INTEN register.	0

Table 263: A/D Flags register (FLAGS, address 0x4001 C068) bit description

Bit	Symbol	Description	Reset value
29	SEQB_INT	Sequence A interrupt/DMA flag. If the MODE bit in the SEQB_CTRL register is 0, this flag will mirror the DATAVALID bit in the sequence A global data register (SEQB_GDAT), which is set at the end of every A/D conversion performed as part of sequence B. It will be cleared automatically when the SEQB_GDAT register is read. If the MODE bit in the SEQB_CTRL register is 1, this flag will be set upon completion of an entire B sequence. In this case it must be cleared by writing a 1 to this SEQB_INT bit. This interrupt must be enabled in the INTEN register.	0
30	THCMP_INT	Threshold Comparison Interrupt/DMA flag. This bit will be set if any of the 12 THCMP flags in the lower bits of this register are set to 1 (due to an enabled out-of-range or threshold-crossing event on any channel). Each type of threshold comparison interrupt on each channel must be individually enabled in the INTEN register to cause this interrupt. This bit will be cleared when all of the component flags in bits 11:0 are cleared via writing 1s to those bits.	0
31	OVR_INT	Overrun Interrupt flag. Any overrun bit in any of the individual channel data registers will cause this interrupt. In addition, if the MODE bit in either of the SEQn_CTRL registers is 0 then the OVERRUN bit in the corresponding SEQn_GDAT register will also cause this interrupt. This interrupt must be enabled in the INTEN register. This bit will be cleared when all of the individual overrun bits have been cleared via reading the corresponding data registers.	0

16.6.11 A/D trim register

The A/D trim register configures the ADC for the appropriate operating range of the analog supply voltage VDDA.

Remark: Failure to set the VRANGE bit correctly causes the ADC to return incorrect conversion results.

Table 264: A/D Flags register (TRM, addresses 0x4001 C06C) bit description

Bit	Symbol	Value	Description	Reset value
4:0	-		Reserved.	-
5	VRANGE		Reserved.	0
		0	High voltage. VDDA = 2.7 V to 3.6 V.	
		1	Low voltage. VDDA = 2.4 V to 2.7 V.	
31:6	-		Reserved.	-

16.7 Functional description

16.7.1 Conversion Sequences

A conversion sequence is a single pass through a series of A/D conversions performed on a selected set of A/D channels. Software can set-up two independent conversion sequences, either of which can be triggered by software or by a transition on one of the hardware triggers. Each sequence can be triggered by a different hardware trigger. One of these conversion sequences is referred to as the A sequence and the other as the B sequence. It is not necessary to employ both sequences.

An optional single-step mode allows advancing through the channels of a sequence one at a time on each successive occurrence of a trigger.

The user can select whether a trigger on the B sequence can interrupt an already-in-progress A sequence. The B sequence, however, can never be interrupted by an A trigger.

16.7.2 Hardware-triggered conversion

Software can select which of these hardware triggers will launch each conversion sequence and it can specify the active edge for the selected trigger independently for each conversion sequence.

For each conversion sequence, if a designated trigger event occurs, one single cycle through that conversion sequence will be launched unless:

- The BURST bit in the ADSEQn_CTRL register for this sequence is set to 1.
- The requested conversion sequence is already in progress.
- A set of conversions for the alternate conversion sequence is already in progress except in the case of a B trigger interrupting an A sequence if the A sequence is set to LOWPRIO.

If any of these conditions is true, the new trigger event will be ignored and will have no effect.

In addition, if the single-step bit for a sequence is set, each new trigger will cause a single conversion to be performed on the next channel in the sequence rather than launching a pass through the entire sequence.

If the A sequence is enabled to be interrupted (i.e. the LOWPRIO bit in the SEQA_CTRL register is set) and a B trigger occurs while an A sequence is in progress, then the following will occur:

- The A/D conversion which is currently in-progress will be aborted.
- The A sequence will be paused, and the B sequence will immediately commence.
- The interrupted A sequence will resume after the B sequence completes, beginning with the conversion that was aborted when the interruption occurred. The channel for that conversion will be re-sampled.

16.7.2.1 Avoiding spurious hardware triggers

Care should be taken to avoid generating a spurious trigger when writing to the SEQn_CTRL register to change the trigger selected for the sequence, switch the polarity of the selected trigger, or to enable the sequence for operation.

In general, the TRIGGER and TRIGPOL bits in the SEQn_CTRL register should only be written to when the sequence is disabled (while the SEQn_ENA bit is LOW). The SEQn_ENA bit itself should only be set when the selected trigger input is in its INACTIVE state (as designated by the TRIGPOL bit). If this condition is not met, a trigger will be generated immediately upon enabling the sequence - even though no actual transition has occurred on the trigger input.

16.7.3 Software-triggered conversion

There are two ways that software can trigger a conversion sequence:

1. **Start Bit:** The first way to software-trigger an sequence is by setting the START bit in the corresponding SEQn_CTRL register. The response to this is identical to occurrence of a hardware trigger on that sequence. Specifically, one cycle of conversions through that conversion sequence will be immediately triggered except as indicated above.
2. **Burst Mode**: The other way to initiate conversions is to set the BURST bit in the SEQn_CTRL register. As long as this bit is 1 the designated conversion sequence will be continuously and repetitively cycled through. Any new software or hardware trigger on this sequence will be ignored.

If a bursting A sequence is allowed to be interrupted (i.e. the LOWPRIO bit in its SEQA_CTRL register is set to 1 and a software or hardware trigger for the B sequence occurs, then the burst will be immediately interrupted and a B sequence will be initiated. The interrupted A sequence will resume continuous cycling, starting with the aborted conversion, after the alternate sequence has completed.

16.7.4 Interrupts

There are four interrupts that can be generated by the ADC:

- Conversion-Complete or Sequence-Complete interrupts for sequences A and B
- Threshold-Compare Out-of-Range Interrupt
- Data Overrun Interrupt

Any of these interrupt requests may be individually enabled or disabled in the INTEN register.

16.7.4.1 Conversion-Complete or Sequence-Complete interrupts

For each of the two sequences, an interrupt request can either be asserted at the end of each A/D conversion performed as part of that sequence or when the entire sequence of conversions is completed. The MODE bits in the SEQn_CTRL registers select between these alternative behaviors.

If the MODE bit for a sequence is 0 (conversion-complete mode) then the interrupt flag for that sequence will reflect the state of the DATAVALID bit in the global data register (SEQn_GDAT) for that sequence. In this case, reading the SEQn_GDAT register will automatically clear the interrupt request.

If the MODE bit for the sequence is 1 (sequence-complete mode) then the interrupt flag must be written-to by software to clear it (except when used as a DMA trigger, in which case it will be cleared in hardware by the DMA engine).

16.7.4.2 Threshold-Compare Out-of-Range Interrupt

Every conversion performed on any channel is automatically compared against a designated set of low and high threshold levels specified in the THRn_HIGH and THRn_LOW registers. The results of this comparison on any individual channel(s) can be enabled to cause a threshold-compare interrupt if that result was above or below the range specified by the two thresholds or, alternatively, if the result represented a crossing of the low threshold in either direction.

This flag must be cleared by a software write to clear the individual THCMP flags in the FLAGS register.

16.7.4.3 Data Overrun Interrupt

This interrupt request will be asserted if any of the OVERRUN bits in the individual channel data registers are set. In addition, the OVERRUN bits in the two sequence global data (SEQn_GDAT) registers will cause this interrupt request IF the MODE bit for that sequence is set to 0 (conversion-complete mode).

This flag will be cleared when the OVERRUN bit that caused it is cleared via reading the register containing it.

Note that the OVERRUN bits in the individual data registers are cleared when data related to that channel is read from either of the global data registers as well as when the individual data registers themselves are read.

16.7.5 Optional operating modes

The following optional modes of A/D operation may be selected in the CTRL register:

Low-power mode. When this mode is selected, the analog portions of the ADC are automatically shut down when no conversions are in progress. The ADC is automatically restarted whenever any hardware or software trigger event occurs. This mode can save an appreciable amount of power when the ADC is not in continuous use, but at the expense of a delay between the trigger event and the onset of sampling and conversion.

16.7.6 DMA control

The sequence-A or sequence-B conversion/sequence-complete interrupts may also be used to generate a DMA trigger. To trigger a DMA transfer, the same conditions must be met as the conditions for generating an interrupt (see Section 16.7.4 and Section 16.6.9).

Remark: If the DMA is used, the ADC interrupt must be disabled in the NVIC.

For DMA transfers, only burst requests are supported. The burst size can be set to one in the DMA channel control register (see Table 146). If the number of ADC channels is not equal to one of the other DMA-supported burst sizes (applicable DMA burst sizes are 1, 4, 8), set the burst size to one.

The DMA transfer size determines when a DMA interrupt is generated. The transfer size can be set to the number of ADC channels being converted. Non-contiguous channels can be transferred by the DMA using the scatter/gather linked lists.

16.7.7 Hardware Trigger Source Selection

Each ADC has a selection of several on-chip and off-chip hardware trigger sources (see Section 16.3.3 “ADC hardware trigger inputs”). The trigger to be used for each conversion sequence is specified in the TRIGGER fields in the two SEQn_CTRL registers.

UM10732

Chapter 17: LPC11U6x/Ex Temperature sensor

Rev. 1.8 — 19 December 2016 **User manual**

17.1 How to read this chapter

The temperature sensor is identical for all parts.

17.2 Basic configuration

Enable power to the temperature sensor by setting the TS_PD bit to zero in the PDRUNCFG register (Table 69).

The calibration of the temperature sensor is documented in the data sheet. See Ref. 4 and Ref. 6.

17.3 Features

- Output linear over device temperature range
- Low power consumption
- Virtually V_{DD}-independent over device voltage range

17.4 General description

The temperature sensor outputs an analog voltage that varies inversely with device temperature over the allowed range. Whenever the temperature sensor is powered, the sensor output is monitored by ADC channel 0.

The only control bit associated with the temperature sensor is the power control bit in the SYSCON block (Table 69). The temperature sensor output is available to the A/D Converter channel 0.

For an accurate measurement of the temperature sensor by the ADC, the ADC must be configured in single-channel burst mode. The last value of a nine-conversion (or more) burst provides an accurate result.

After the Temperature Sensor is powered, it requires some time to stabilize and output a voltage that correctly represents the device temperature. A much shorter settling time applies after switching the A/D converter to use the sensor output. Software can deal with both of these factors by repeatedly converting and reading the Temperature Sensor output via the A/D converter until a consistent result is obtained.

17.5 Register description

The temperature sensor has no user-configurable registers except for the power control located in the in the System Control block. See Table 69.

18.1 How to read this chapter

The number of pinned out SCTimer/PWM inputs and outputs depends on the package size.

Table 265. Available SCT input and output pin functions

Package	**SCT0**		**SCT1**	
	Inputs	**Outputs**	**Inputs**	**Outputs**
LQFP48	SCT0_IN1	SCT0_OUT1, SCT0_OUT2, SCT0_OUT3	-	-
LQFP64	SCT0_IN1, SCT0_IN2, SCT0_IN3	SCT0_OUT0, SCT0_OUT1, SCT0_OUT2, SCT0_OUT3	SCT1_IN3	SCT1_OUT2, SCT1_OUT3
LQFP100	SCT0_IN0, SCT0_IN1, SCT0_IN2, SCT0_IN3	SCT0_OUT0, SCT0_OUT1, SCT0_OUT2, SCT0_OUT3	SCT1_IN0, SCT1_IN1, SCT1_IN2, SCT1_IN3	SCT1_OUT0, SCT1_OUT1, SCT1_OUT2, SCT1_OUT3

18.2 Features

- Each SCTimer/PWM supports:
  - 5 match/capture registers.
  - 6 events.
  - 8 states.
  - 4 inputs and 4 outputs.
- Counter/timer features:
  - Each SCTimer is configurable as two 16-bit counters or one 32-bit counter.
  - Counters clocked by system clock or selected input.
  - Configurable as up counters or up-down counters.
  - Configurable number of match and capture registers. Up to five match and capture registers total.
  - Upon match and/or an input or output transition create the following events: interrupt; stop, limit, halt the timer or change counting direction; toggle outputs; change the state.
  - Counter value can be loaded into capture register triggered by a match or input/output toggle.
- PWM features:
  - Counters can be used in conjunction with match registers to toggle outputs and create time-proportioned PWM signals.

  - Up to 4 single-edge or dual-edge PWM outputs with independent duty cycle and common PWM cycle length.
- Event creation features:
  - The following conditions define an event: a counter match condition, an input (or output) condition such as an rising or falling edge or level, a combination of match and/or input/output condition.
  - Selected events can limit, halt, start, or stop a counter or change its direction.
  - Events trigger state changes, output toggles, interrupts, and DMA transactions.
  - Match register 0 can be used as an automatic limit.
  - In bi-directional mode, events can be enabled based on the count direction.
  - Match events can be held until another qualifying event occurs.
- State control features:
  - A state is defined by events that can happen in the state while the counter is running.
  - A state changes into another state as a result of an event.
  - Each event can be assigned to one or more states.
  - State variable allows sequencing across multiple counter cycles.

18.3 Basic configuration

Configure the SCT0/1 as follows:

- SCTimer0 and SCTimer1 (SCT0/1) share one bit in the SYSAHBCLKCTRL register (Table 40) to enable the clock to the SCT register interfaces and the peripheral clocks.
- Clear the SCT0/1 peripheral resets using the PRESETCTRL register (Table 23).
- The SCT0 and SCT1 combined interrupts are ORed in slot #13 in the NVIC (Table 6).
- Use the IOCON registers to connect the SCT inputs and outputs to external pins. See Table 83.
- The SCT DMA request lines are connected to the DMA trigger inputs via the DMA_ITRIG_PINMUX registers. See Table 148 “DMA trigger input mux registers 0 to 15 (DMA_ITRIG_INMUX[0:15], address 0x4002 80E0 (DMA_ITRIG_INMUX0) to 0x4002 811C (DMA_ITRIG_INMUX15)) bit description”.

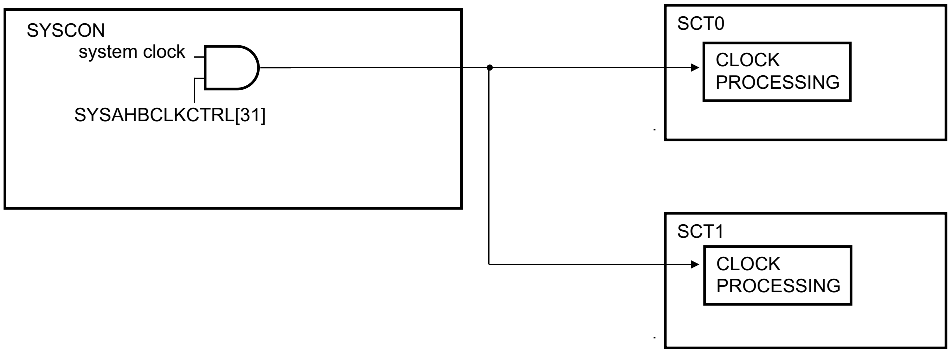


Fig 62. SCT clocking

18.4 Pin description

Table 266. SCT pin description

Pin	Type	Description
SCT0_IN0, SCT0_IN1, SCT0_IN2, SCT0_IN3	Input	SCTimer0 capture and event inputs.
SCT0_OUT0, SCT0_OUT1, SCT0_OUT2, SCT0_OUT3	Output	SCTimer0 match and PWM outputs. SCT0_OUT0 is ORed with the Match output 1 of CT32B0 for one of the possible ADC input triggers.
SCT1_IN0, SCT1_IN1, SCT1_IN2, SCT1_IN3	Input	SCTimer1 capture and event inputs.
SCT1_OUT0, SCT1_OUT1, SCT1_OUT2, SCT1_OUT3	Output	SCTimer1 match and PWM outputs. SCT1_OUT0 is ORed with the Match output 1 of CT16B0 for one of the possible the ADC input triggers.

18.5 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.

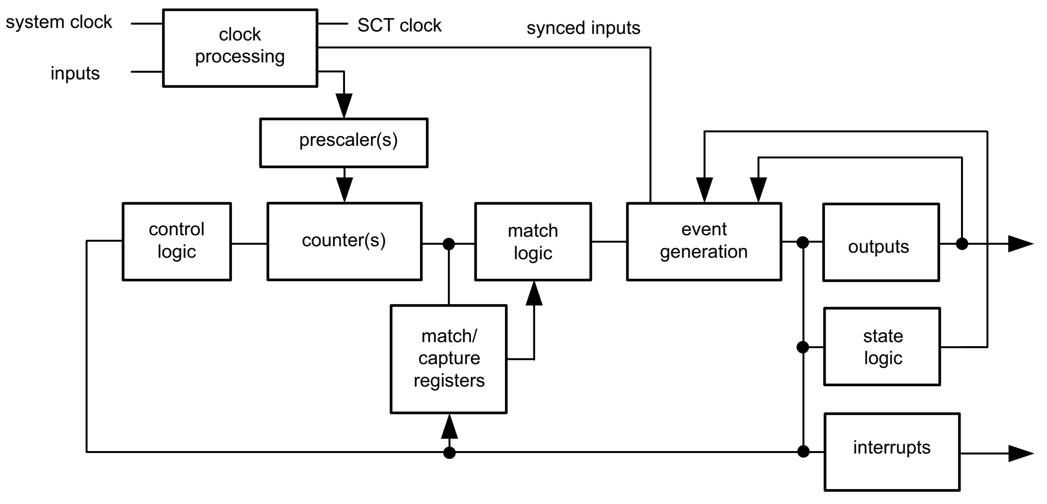


Fig 63. SCTimer/PWM block diagram

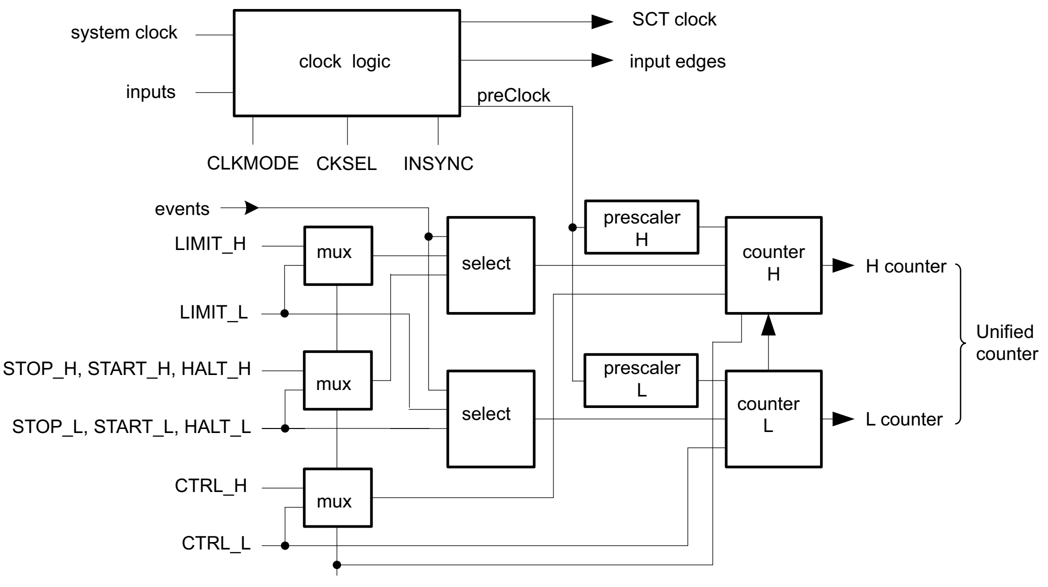


Fig 64. SCTimer/PWM counter and select logic

18.6 Register description

The register addresses of the State Configurable Timer are shown in Table 267. 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 capture and capture control registers.
   - REGMODEn = 0: Registers operate as match and reload registers.

Table 267. Register overview: State Configurable Timer (base address 0x5000 C000 (SCT0) and 0x5000 E000 (SCT1))

Name	Access	Address offset	Description	Reset value	Reference
CONFIG	R/W	0x000	SCT configuration register	0x0000 7E00	Table 268
CTRL	R/W	0x004	SCT control register	0x0004 0004	Table 269
CTRL_L	R/W	0x004	SCT control register low counter 16-bit	0x0004 0004	Table 269
CTRL_H	R/W	0x006	SCT control register high counter 16-bit	0x0004 0004	Table 269
LIMIT	R/W	0x008	SCT limit register	0x0000 0000	Table 270
LIMIT_L	R/W	0x008	SCT limit register low counter 16-bit	0x0000 0000	Table 270
LIMIT_H	R/W	0x00A	SCT limit register high counter 16-bit	0x0000 0000	Table 270
HALT	R/W	0x00C	SCT halt condition register	0x0000 0000	Table 271
HALT_L	R/W	0x00C	SCT halt condition register low counter 16-bit	0x0000 0000	Table 271
HALT_H	R/W	0x00E	SCT halt condition register high counter 16-bit	0x0000 0000	Table 271
STOP	R/W	0x010	SCT stop condition register	0x0000 0000	Table 272
STOP_L	R/W	0x010	SCT stop condition register low counter 16-bit	0x0000 0000	Table 272
STOP_H	R/W	0x012	SCT stop condition register high counter 16-bit	0x0000 0000	Table 272
START	R/W	0x014	SCT start condition register	0x0000 0000	Table 273
START_L	R/W	0x014	SCT start condition register low counter 16-bit	0x0000 0000	Table 273
START_H	R/W	0x016	SCT start condition register high counter 16-bit	0x0000 0000	Table 273
-	-	0x018 - 0x03C	Reserved		-
COUNT	R/W	0x040	SCT counter register	0x0000 0000	Table 274
COUNT_L	R/W	0x040	SCT counter register low counter 16-bit	0x0000 0000	Table 274
COUNT_H	R/W	0x042	SCT counter register high counter 16-bit	0x0000 0000	Table 274
STATE	R/W	0x044	SCT state register	0x0000 0000	Table 275
STATE_L	R/W	0x044	SCT state register low counter 16-bit	0x0000 0000	Table 275
STATE_H	R/W	0x046	SCT state register high counter 16-bit	0x0000 0000	Table 275
INPUT	RO	0x048	SCT input register	0x0000 0000	Table 276
REGMODE	R/W	0x04C	SCT match/capture registers mode register	0x0000 0000	Table 277
REGMODE_L	R/W	0x04C	SCT match/capture registers mode register low counter 16-bit	0x0000 0000	Table 277
REGMODE_H	R/W	0x04E	SCT match/capture registers mode register high counter 16-bit	0x0000 0000	Table 277
OUTPUT	R/W	0x050	SCT output register	0x0000 0000	Table 278

Table 267. Register overview: State Configurable Timer (base address 0x5000 C000 (SCT0) and 0x5000 E000 (SCT1)) *…continued*

Name	Access	Address offset	Description	Reset value	Reference
OUTPUTDIRCTRL	R/W	0x054	SCT output counter direction control register	0x0000 0000	Table 279
RES	R/W	0x058	SCT conflict resolution register	0x0000 0000	Table 280
DMAREQ0	R/W	0x05C	SCT DMA request 0 register	0x0000 0000	Table 281
DMAREQ1	R/W	0x060	SCT DMA request 1 register	0x0000 0000	Table 282
-	-	0x064 - 0x0EC	Reserved	-	-
EVEN	R/W	0x0F0	SCT event enable register	0x0000 0000	Table 283
EVFLAG	R/W	0x0F4	SCT event flag register	0x0000 0000	Table 284
CONEN	R/W	0x0F8	SCT conflict enable register	0x0000 0000	Table 285
CONFLAG	R/W	0x0FC	SCT conflict flag register	0x0000 0000	Table 286
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 286
MATCH0_L to MATCH4_L	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	0x0000 0000	Table 286
MATCH0_H to MATCH4_H	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	0x0000 0000	Table 286
CAP0 to CAP4	R/W	0x100 to 0x110	SCT capture register of capture channel 0 to 4; REGMOD0 to REGMODE4 = 1	0x0000 0000	Table 288
CAP0_L to CAP4_L	R/W	0x100 to 0x110	SCT capture register of capture channel 0 to 4; low counter 16-bit; REGMOD0_L to REGMODE4_L = 1	0x0000 0000	Table 288
CAP0_H to CAP4_H	R/W	0x102 to 0x112	SCT capture register of capture channel 0 to 4; high counter 16-bit; REGMOD0_H to REGMODE4_H = 1	0x0000 0000	Table 288
MATCHREL0 to MATCHREL4	R/W	0x200 to 0x210	SCT match reload value register 0 to 4; REGMOD0 = 0 to REGMODE4 = 0	0x0000 0000	Table 289
MATCHREL0_L to MATCHREL4_L	R/W	0x200 to 0x210	SCT match reload value register 0 to 4; low counter 16-bit; REGMOD0_L = 0 to REGMODE7_L = 0	0x0000 0000	Table 289
MATCHREL0_H to MATCHREL4_H	R/W	0x202 to 0x212	SCT match reload value register 0 to 4; high counter 16-bit; REGMOD0_H = 0 to REGMODE4_H = 0	0x0000 0000	Table 289
CAPCTRL0 to CAPCTRL4	R/W	0x200 to 0x210	SCT capture control register 0 to 4; REGMOD0 = 1 to REGMODE4 = 1	0x0000 0000	Table 290
CAPCTRL0_L to CAPCTRL4_L	R/W	0x200 to 0x210	SCT capture control register 0 to 4; low counter 16-bit; REGMOD0_L = 1 to REGMODE4_L = 1	0x0000 0000	Table 290
CAPCTRL0_H to CAPCTRL4_H	R/W	0x202 to 0x212	SCT capture control register 0 to 4; high counter 16-bit; REGMOD0 = 1 to REGMODE4 = 1	0x0000 0000	Table 290
EV0_STATE	R/W	0x300	SCT event state register 0	0x0000 0000	Table 291
EV0_CTRL	R/W	0x304	SCT event control register 0	0x0000 0000	Table 292
EV1_STATE	R/W	0x308	SCT event state register 1	0x0000 0000	Table 291
EV1_CTRL	R/W	0x30C	SCT event control register 1	0x0000 0000	Table 292

Table 267. Register overview: State Configurable Timer (base address 0x5000 C000 (SCT0) and 0x5000 E000 (SCT1)) *…continued*

Name	Access	Address offset	Description	Reset value	Reference
EV2_STATE	R/W	0x310	SCT event state register 2	0x0000 0000	Table 291
EV2_CTRL	R/W	0x314	SCT event control register 2	0x0000 0000	Table 292
EV3_STATE	R/W	0x318	SCT event state register 3	0x0000 0000	Table 291
EV3_CTRL	R/W	0x31C	SCT event control register 3	0x0000 0000	Table 292
EV4_STATE	R/W	0x320	SCT event state register 4	0x0000 0000	Table 291
EV4_CTRL	R/W	0x324	SCT event control register4	0x0000 0000	Table 292
EV5_STATE	R/W	0x328	SCT event state register 5	0x0000 0000	Table 291
EV5_CTRL	R/W	0x32C	SCT event control register 5	0x0000 0000	Table 292
OUT0_SET	R/W	0x500	SCT output 0 set register	0x0000 0000	Table 293
OUT0_CLR	R/W	0x504	SCT output 0 clear register	0x0000 0000	Table 294
OUT1_SET	R/W	0x508	SCT output 1 set register	0x0000 0000	Table 293
OUT1_CLR	R/W	0x50C	SCT output 1 clear register	0x0000 0000	Table 294
OUT2_SET	R/W	0x510	SCT output 2 set register	0x0000 0000	Table 293
OUT2_CLR	R/W	0x514	SCT output 2 clear register	0x0000 0000	Table 294
OUT3_SET	R/W	0x518	SCT output 3 set register	0x0000 0000	Table 293
OUT3_CLR	R/W	0x51C	SCT output 3 clear register	0x0000 0000	Table 294

18.6.1 SCT configuration register

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

Table 268. SCT configuration register (CONFIG, address 0x5000 C000 (SCT0) and 0x5000 E000 (SCT1)) bit description

Bit	Symbol	Value	Description	Reset value
0	UNIFY		SCT operation	0
		0	The SCT operates as two 16-bit counters named L and H.	
		1	The SCT operates as a unified 32-bit counter.	
2:1	CLKMODE		SCT clock mode	00
		0x0	The bus clock clocks the SCT and prescalers.	
		0x1	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	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	Prescaled SCT input. The SCT and prescalers are clocked by the input edge selected by the CKSEL field. In this mode, most of the SCT is clocked by the (selected polarity of the) input. The outputs are switched synchronously to the input clock. The input clock rate must be at least half the system clock rate and can the same or faster than the system clock.	

Table 268. SCT configuration register (CONFIG, address 0x5000 C000 (SCT0) and 0x5000 E000 (SCT1)) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
6:3	CKSEL		SCT clock select	0000
		0x0	Rising edges on input 0.	
		0x1	Falling edges on input 0.	
		0x2	Rising edges on input 1.	
		0x3	Falling edges on input 1.	
		0x4	Rising edges on input 2.	
		0x5	Falling edges on input 2.	
		0x6	Rising edges on input 3.	
		0x7	Falling edges on input 3.	
7	NORELAOD_L	-	A 1 in this bit prevents the lower match registers from being reloaded from their respective reload registers. 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
16:9	INSYNC	-	Synchronization for input N (bit 9 = input 0, bit 10 = input 1,..., bit 12 = input 3); all other bits are reserved. A 1 in one of these bits subjects the corresponding input to synchronization to the SCT clock, before it is used to create an event. If an input is synchronous to the SCT clock, keep its bit 0 for faster response. When the CKMODE 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	-

18.6.2 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.)

Remark: If CLKMODE = 0x3 is selected, wait at least 12 system clock cycles between a write access to the H, L or unified version of this register and the next write access. This restriction does not apply when writing to the HALT bit or bits and then writing to the CTRL register again to restart the counters - for example because software must update the MATCH register, which is only allowed when the counters are halted.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 269. SCT control register (CTRL, address 0x5000 C004 (SCT0) and 0x5000 E004 (SCT1)) 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
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	The counter counts up to its limit condition, then is cleared to zero.	
		1	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

Table 269. SCT control register (CTRL, address 0x5000 C004 (SCT0) and 0x5000 E004 (SCT1)) bit description

Bit	Symbol	Value	Description	Reset value
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	The H counter counts up to its limit condition, then is cleared to zero.	
		1	The H counter counts up to its limit, then counts down to a limit condition or to 0.	
28:21	PRE_H	-	Specifies the factor by which the SCT clock is prescaled to produce the H counter clock. The counter clock is clocked at the rate of the SCT clock divided by PRELH+1. **Remark:** Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.	0
31:29	-		Reserved	

18.6.3 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 268).

Table 270. SCT limit register (LIMIT, address 0x5000 C008 (SCT0) and 0x5000 E008 (SCT1)) 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.	-

18.6.4 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 269).

Table 271. SCT halt condition register (HALT, address 0x5000 C00C (SCT0) and 0x5000 E00C (SCT1)) 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.	-

18.6.5 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 272. SCT stop condition register (STOP, address 0x5000 C010 (SCT0) and 0x5000 E010 (SCT1)) 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.	-

18.6.6 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 273. SCT start condition register (START, address 0x5000 C014 (SCT0) and 0x5000 E014 (SCT1)) 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.	-

18.6.7 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.

Writing to the COUNT_L, COUNT_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). Software can read the counter registers at any time.

Table 274. SCT counter register (COUNT, address 0x5000 C040 (SCT0) and 0x5000 E040 (SCT1)) 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

18.6.8 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 to the STATE_L, STATE_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register).

The state variable is the main feature that distinguishes the 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 and DMA requests
- modify the state variable

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

A state variable can be used to track and control multiple cycles of the associated counter in any desired operational sequence. The state variable is logically associated with a state machine diagram which represents the SCT configuration. See Section 18.6.23 and 18.6.24 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 275. SCT state register (STATE, address 0x5000 C044 (SCT0) and 0x5000 E044 (SCT1)) 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.	

18.6.9 SCT input register

Software can read the state of the SCT inputs in this read-only register in slightly different forms.

1. The AIN bit represents the input sampled by the SCT clock. This corresponds to a nearly direct read-out of the input but can cause spurious fluctuations in case of an asynchronous input signal.
2. The SIN bit represents the input sampled by the SCT clock after the INSYNC select (this signal is also used for event generation):
   - If the INSYNC bit is set for the input, the input is synchronized to the SCT clock using three SCT clock cycles resulting in a stable signal that is delayed by three SCT clock cycles.
   - If the INSYNC bit is not set, the SIN bit value is the same as the AIN bit value.

Table 276. SCT input register (INPUT, address 0x5000 C048 (SCT0) and 0x5000 E048 (SCT1)) bit description

Bit	Symbol	Description	Reset value
0	AIN0	. Input 0 state.Direct read.	-
1	AIN1	Input 1 state. Direct read.	-
2	AIN2	Input 2 state. Direct read.	-
3	AIN3	Input 3 state. Direct read.	-
15:4	-	Reserved.	-

Table 276. SCT input register (INPUT, address 0x5000 C048 (SCT0) and 0x5000 E048 (SCT1)) bit description

Bit	Symbol	Description	Reset value
16	SIN0	Input 0 state.	-
17	SIN1	Input 1 state.	-
18	SIN2	Input 2 state.	-
19	SIN3	Input 3 state.	-
31:20	-	Reserved	-

18.6.10 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 18.6.19) or as a Capture register (see Section 18.6.20). Each Match/Capture register has an accompanying register which serves as a Reload register when the register is used as a Match register (Section 18.6.21) or as a Capture-Control register when the register is used as a capture register (Section 18.6.22). REGMODE_H is used only when the UNIFY bit is 0.

Table 277. SCT match/capture registers mode register (REGMODE, address 0x5000 C04C (SCT0) and 0x5000 E04C (SCT1)) 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.	-

18.6.11 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 the OUT register is only allowed when all counters (L-counter, H-counter, and unified counter) are halted (HALT bits are set to 1 in the CTRL register).

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

Table 278. SCT output register (OUTPUT, address 0x5000 C050 (SCT0) and 0x5000 E050 (SCT1)) 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	

18.6.12 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 18.6.25 and Section 18.6.26).

Table 279. SCT bidirectional output control register (OUTPUTDIRCTRL, address 0x5000 C054 (SCT0) and 0x5000 E054 (SCT1)) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SETCLR0		Set/clear operation on output 0. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
3:2	SETCLR1		Set/clear operation on output 1. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
5:4	SETCLR2		Set/clear operation on output 2. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
7:6	SETCLR3		Set/clear operation on output 3. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
31:8	-		Reserved	-

18.6.13 SCT conflict resolution register

The registers OUTn_SET (Section 18.6.25) and OUTn_CLR (Section 18.6.26) 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 280. SCT conflict resolution register (RES, address 0x5000 C058 (SCT0) and 0x5000 E058 (SCT1)) 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.	
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	-

18.6.14 SCT DMA request 0 and 1 registers

The SCT includes two DMA request outputs. These registers enable the DMA requests to be triggered when a particular event occurs or when counter Match registers are loaded from its Reload registers.

Event-triggered DMA requests are particularly useful for launching DMA activity to or from other peripherals under the control of the SCT.

Table 281. SCT DMA 0 request register (DMAREQ0, address 0x5000 C05C (SCT0) and 0x5000 E05C (SCT1)) bit description

Bit	Symbol	Description	Reset value
5:0	DEV_0	If bit n is one, event n sets DMA request 0 (event 0 = bit 0, event 1 = bit 1,..., event 5 = bit 5).	0

Table 281. SCT DMA 0 request register (DMAREQ0, address 0x5000 C05C (SCT0) and 0x5000 E05C (SCT1)) bit description

Bit	Symbol	Description	Reset value
29:6	-	Reserved	-
30	DRL0	A 1 in this bit makes the SCT set DMA request 0 when it loads the Match_L/Unified registers from the Reload_L/Unified registers.	
31	DRQ0	This read-only bit indicates the state of DMA Request 0. Note that if the related DMA channel is enabled and properly set up, it is unlikely that software will see this flag, it will be cleared rapidly by the DMA service. The flag remaining set could point to an issue with DMA setup.	

Table 282. SCT DMA 1 request register (DMAREQ1, address 0x5000 C060 (SCT0) and 0x5000 E060 (SCT1)) bit description

Bit	Symbol	Description	Reset value
5:0	DEV_1	If bit n is one, event n sets DMA request 1 (event 0 = bit 0, event 1 = bit 1,..., event 5 = bit 5).	0
29:6	-	Reserved	-
30	DRL1	A 1 in this bit makes the SCT set DMA request 1 when it loads the Match L/Unified registers from the Reload L/Unified registers.	
31	DRQ1	This read-only bit indicates the state of DMA Request 1. Note that if the related DMA channel is enabled and properly set up, it is unlikely that software will see this flag, it will be cleared rapidly by the DMA service. The flag remaining set could point to an issue with DMA setup.	

18.6.15 SCT flag enable register

This register enables flags to request an interrupt if the FLAGn bit in the SCT event flag register (Section 18.6.16) is also set.

Table 283. SCT flag enable register (EVEN, address 0x5000 C0F0 (SCT0) and 0x5000 E0F0 (SCT1)) 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	

18.6.16 SCT event flag register

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

Table 284. SCT event flag register (EVFLAG, address 0x5000 C0F4 (SCT0) and 0x5000 E0F4 (SCT1)) 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	-

18.6.17 SCT conflict enable register

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

Table 285. SCT conflict enable register (CONEN, address 0x5000 C0F8 (SCT0) and 0x5000 E0F8 (SCT1)) 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	

18.6.18 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 286. SCT conflict flag register (CONFLAG, address 0x5000 C0FC (SCT0) and 0x5000 E0FC (SCT1)) 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

18.6.19 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 the MATCH_L, MATCH_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). Match events occur in the SCT clock in which the counter is (or would be) incremented to the next value. When a Match event limits its counter as described in Section 18.6.3, 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 287. SCT match registers 0 to 4 (MATCH[0:4], address 0x5000 C100 (MATCH0) to 0x5000 C110 (MATCH4) (SCT0) and address 0x5000 E100 (MATCH0) to 0x50004E10 (MATCH4) (SCT1)) bit description (REGMODEn bit = 0)

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

18.6.20 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 288. SCT capture registers 0 to 4 (CAP[0:4], address 0x5000 C100 (CAP0) to 0x5000 C110 (CAP4) (SCT0) and address 0x5000 E100 (CAP0) to 0x5000 E110 (CAP4) (SCT1)) bit description (REGMODEn bit = 1)

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

18.6.21 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 289. SCT match reload registers 0 to 4 (MATCHREL[0:4], address 0x5000 C200 (MATCHREL0) to 0x5000 C210 (MATCHREL4) (SCT0) and 0x5000 E200 (MATCHREL0) to 0x5000 E210 (MATCHREL4) (SCT1)) bit description (REGMODEn bit = 0)

Bit	Symbol	Description	Reset value
15:0	RELOADn_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	RELOADn_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

18.6.22 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 290. SCT capture control registers 0 to 4 (CAPCTRL[0:4], address 0x5000 C200 (CAPCTRL0) to 0x5000 C210 (CAPCTRL4) (SCT0) and 0x5000 E200 (CAPCTRL0) to 0x5000 E210 (CAPCTRL4) (SCT1)) bit description (REGMODEn bit = 1)

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

18.6.23 SCT event state 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 EVCTRLn 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 effectively permanently state-enables this event.

Table 291. SCT event state mask registers 0 to 5 (EV[0:5]_STATE, addresses 0x5000 C300 (EV0_STATE) to 0x5000 C328 (EV5_STATE) (SCT0) and 0x5000 E300 (EV0_STATE) to 0x5000 E328 (EV5_STATE) (SCT1)) bit description

Bit	Symbol	Description	Reset value
7:0	STATEMSKn	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,..., state 7 = bit 7).	0
31:8	-	Reserved.	-

18.6.24 SCT event control registers 0 to 5

This register defines the conditions for event n (n = 0 to 5) 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 292. SCT event control register 0 to 5 (EV[0:5]_CTRL, address 0x5000 C304 (EV0_CTRL) to 0x5000 C32C (EV5_CTRL) (SCT0) and 0x5000 E304 (EV0_CTRL) to 0x5000 E32C (EV5_CTRL) (SCT1)) bit description

Bit	Symbol	Value	Description	Reset value
3:0	MATCHSEL	-	Selects the Match register associated with this event (if any). A match can occur only when the counter selected by the HEVENT bit is running.	0
4	HEVENT		Select L/H counter. Do not set this bit if UNIFY = 1.	0
		0	Selects the L state and the L match register selected by MATCHSEL.	
		1	Selects the H state and the H match register selected by MATCHSEL.	
5	OUTSEL		Input/output select	0
		0	Selects the inputs elected by IOSEL.	
		1	Selects the outputs selected by IOSEL.	
9:6	IOSEL	-	Selects the input or output signal number (0 to 3) associated with this event (if any). Do not select an input in this register, if CKMODE is 1x. In this case the clock input is an implicit ingredient of every event.	0

Table 292. SCT event control register 0 to 5 (EV[0:5]_CTRL, address 0x5000 C304 (EV0_CTRL) to 0x5000 C32C (EV5_CTRL) (SCT0) and 0x5000 E304 (EV0_CTRL) to 0x5000 E32C (EV5_CTRL) (SCT1)) bit description

Bit	Symbol	Value	Description	Reset value
11:10	IOCOND		Selects the I/O condition for event n. (The detection of edges on outputs lag the conditions that switch the outputs by one SCT clock). In order to guarantee proper edge/state detection, an input must have a minimum pulse width of at least one SCT clock period .	0
		0x0	LOW	
		0x1	Rise	
		0x2	Fall	
		0x3	HIGH	
13:12	COMBMODE		Selects how the specified match and I/O condition are used and combined.	
		0x0	OR. The event occurs when either the specified match or I/O condition occurs.	
		0x1	MATCH. Uses the specified match only.	
		0x2	IO. Uses the specified I/O condition only.	
		0x3	AND. The event occurs when the specified match and I/O condition occur simultaneously.	
14	STATELD		This bit controls how the STATEV value modifies the state selected by HEVENT when this event is the highest-numbered event occurring for that state.	
		0	STATEV value is added into STATE (the carry-out is ignored).	
		1	STATEV value is loaded into STATE.	
19:15	STATEV		This value is loaded into or added to the state selected by HEVENT, depending on STATELD, when this event is the highest-numbered event occurring for that state. If STATELD and STATEV are both zero, there is no change to the STATE value.	
20	MATCHMEM		If this bit is one and the COMBMODE field specifies a match component to the triggering of this event, then a match is considered to be active whenever the counter value is GREATER THAN OR EQUAL TO the value specified in the match register when counting up, LESS THEN OR EQUAL TO the match value when counting down. If this bit is zero, a match is only be active during the cycle when the counter is equal to the match value.	
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	

18.6.25 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 OUTPUTDIRCTRL register.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 293. SCT output set register (OUT[0:3]_SET, address 0x5000 C500 (OUT0_SET) to 0x5000 C518 (OUT3_SET) (SCT0) and 0x5000 E500 (OUT0_SET) to 0x5000 E518 (OUT3_SET) (SCT1)) 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	

18.6.26 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.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 294. SCT output clear register (OUT[0:3]_CLR, address 0x5000 C504 (OUT0_CLR) to 0x5000 C51C (OUT3_CLR) and 0x5000 E504 (OUT0_CLR) to 0x5000 E51C (OUT3_CLR) (SCT1)) 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	

18.7 Functional description

18.7.1 Match logic

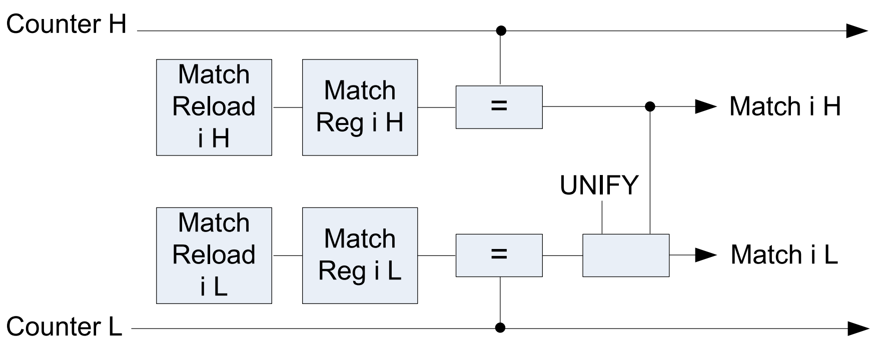


Fig 65. Match logic

18.7.2 Capture logic

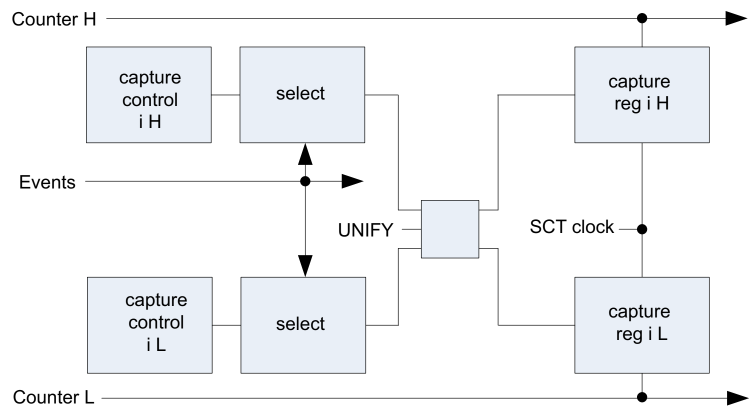


Fig 66. Capture logic

18.7.3 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.

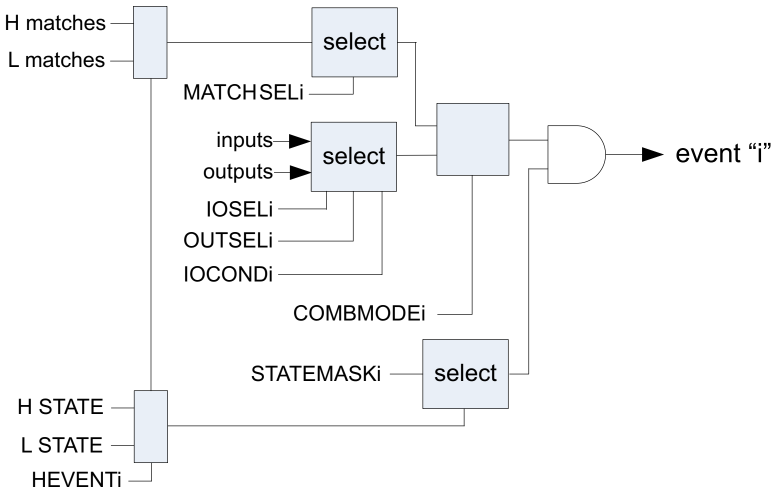


Fig 67. Event selection

18.7.4 Output generation

Figure 68 shows one output slice of the SCT.

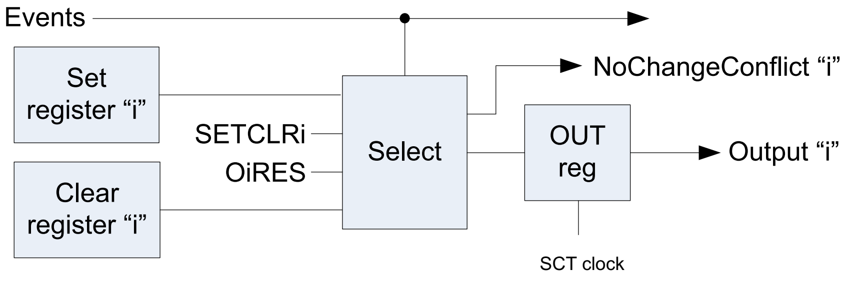


Fig 68. Output slice i

18.7.5 State logic

The SCT can be configured as a timer/counter with multiple programmable states. The states are user-defined through the events that can be captured in each particular state. In a multi-state SCT. the SCT can change from one state to another state when a user-defined event triggers a state change. The state change is triggered through each event’s EV_CTRL register in one of the following ways:

- The event can increment the current state number by a new value.
- The event can write a new state value.

If an event increments the state number beyond the number of available states, the SCT enters a locked state in which all further events are ignored while the counter is still running. Software must interfere to change out of this state.

Software can capture the counter value (and potentially create an interrupt and write to all outputs) when the event moving the SCT into a locked state occurs.Later, while the SCT is in the locked state, software can read the counter again to record the time passed since the locking event and can also read the state variable to obtain the current state number

If the SCT registers an event that forces an abort, putting the SCT in a locked state can be a safe way to record the time that has passed since the abort event while no new events are allowed to occur. Since multiple states (any state number between the maximum implemented state and 31) are locked states, multiple abort or error events can be defined each incrementing the state number by a different value.

18.7.6 Interrupt generation

The SCT generates one interrupt to the NVIC.

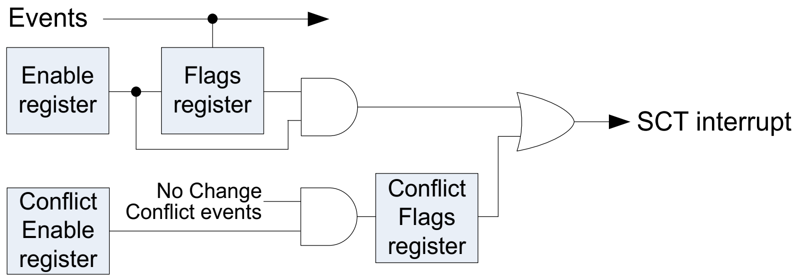


Fig 69. SCT interrupt generation

18.7.7 Clearing the prescaler

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

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

When BIDIR is 0, a limit event caused by an I/O signal can clear a non-zero prescaler. However, a limit event caused by a Match only clears a non-zero prescaler in one special case as described Section 18.7.8.

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

18.7.8 Match vs. I/O events

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

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

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

Table 295 shows when the various kinds of events can occur.

Table 295. 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).

18.7.9 SCTimer/PWM 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 18.7.10.
- To start, run, and stop the SCT, see Section 18.7.11.
- To configure the SCT as simple event controlled counter/timer, see Section 18.7.12.

18.7.10 Configure the SCT

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

18.7.10.1 Configure the counter

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

18.7.10.2 Configure the match and capture registers

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

18.7.10.3 Configure events and event responses

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

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

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

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

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

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

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

18.7.10.4 Configure multiple states

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

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

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

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

18.7.10.5 Miscellaneous options

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

18.7.11 Run the SCT

1. Configure the SCT (see Section 18.7.10 “Configure the SCT”).
2. Write to the STATE register to define the initial state. By default the initial state is state 0.
3. To start the SCT, write to the CTRL register:
   - Clear the counters.
   - Clear or set the STOP_L and/or STOP_H bits.

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

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

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

18.7.12 Configure the SCT without using states

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

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

  In effect, the event is allowed to occur in a single state which never changes while the counter is running.

18.7.13 PWM Example

Figure 70 shows a simple application of the SCT using two sets of match events (EV0/1 and EV3/4) to set/clear SCT output 0. The timer is automatically reset whenever it reaches the MAT0 match value.

In the initial state 0, match event EV0 sets output 0 to HIGH and match event EV1 clears output 0. The SCT input 0 is monitored: If input0 is found LOW by the next time the timer is reset(EV2), the state is changed to state 1, and EV3/4 are enabled, which create the same output but triggered by different match values. If input 0 is found HIGH by the next time the timer is reset, the associated event (EV5) causes the state to change back to state 0where the events EV0 and EV1 are enabled.

The example uses the following SCT configuration:

- 1 input
- 1 output
- 5 match registers
- 6 events and match 0 used with autolimit function
- 2 states

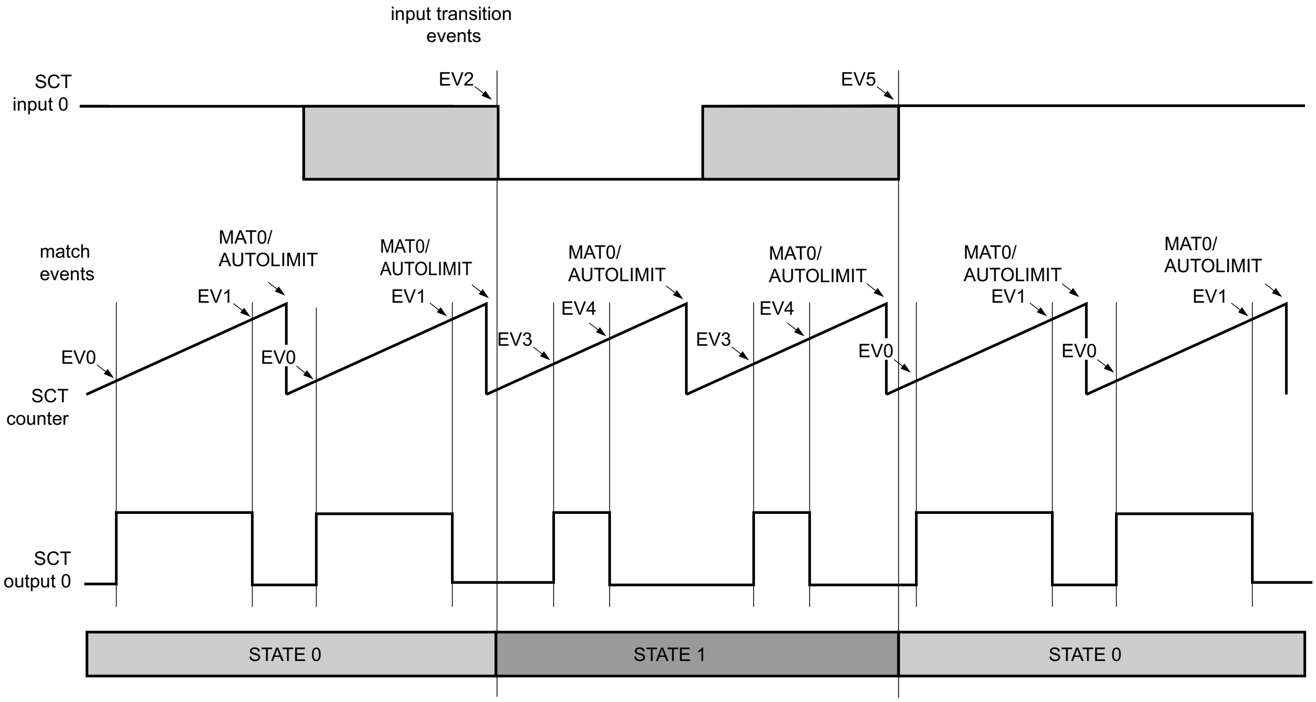


Fig 70. SCT configuration example

This application of the SCT uses the following configuration (all register values not listed in Table 296 are set to their default values):

Table 296. SCT configuration example

Configuration	Registers	Setting
Counter	CONFIG	Uses one counter (UNIFY = 1).
	CONFIG	Enable the autolimit for MAT0. (AUTOLIMIT = 1.)
	CTRL	Uses unidirectional counter (BIDIR_L = 0).
Clock base	CONFIG	Uses default values for clock configuration.
Match/Capture registers	REGMODE	Configure one match register for each match event by setting REGMODE_L bits 0,1, 2, 3, 4 to 0. This is the default.
Define match values	MATCH0/1/2/3/4	Set a match value MATCH0/1/2/3/4_L in each register. The match 0 register serves as an automatic limit event that resets the counter. without using an event. To enable the automatic limit, set the AUTOLIMIT bit in the CONFIG register.
Define match reload values	MATCHREL0/1/2/3/4	Set a match reload value RELOAD0/1/2/3/4_L in each register (same as the match value in this example).
Define when event 0 occurs	EV0_CTRL	• Set COMBMODE = 0x1. Event 0 uses match condition only. • Set MATCHSEL = 1. Select match value of match register 1. The match value of MAT1 is associated with event 0.
Define when event 1 occurs	EV1_CTRL	• Set COMBMODE = 0x1. Event 1 uses match condition only. • Set MATCHSEL = 2 Select match value of match register 2. The match value of MAT2 is associated with event 1.

Table 296. SCT configuration example

Configuration	Registers	Setting
Define when event 2 occurs	EV2_CTRL	• Set COMBMODE = 0x3. Event 2 uses match condition and I/O condition. • Set IOSEL = 0. Select input 0. • Set IOCOND = 0x0. Input 0 is LOW. • Set MATCHSEL = 0. Chooses match register 0 to qualify the event.
Define how event 2 changes the state	EV2_CTRL	Set STATEV bits to 1 and the STATED bit to 1. Event 2 changes the state to state 1.
Define when event 3 occurs	EV3_CTRL	• Set COMBMODE = 0x1. Event 3 uses match condition only. • Set MATCHSEL = 0x3. Select match value of match register 3. The match value of MAT3 is associated with event 3..
Define when event 4 occurs	EV4_CTRL	• Set COMBMODE = 0x1. Event 4 uses match condition only. • Set MATCHSEL = 0x4. Select match value of match register 4.The match value of MAT4 is associated with event 4.
Define when event 5 occurs	EV5_CTRL	• Set COMBMODE = 0x3. Event 5 uses match condition and I/O condition. • Set IOSEL = 0. Select input 0. • Set IOCOND = 0x3. Input 0 is HIGH. • Set MATCHSEL = 0. Chooses match register 0 to qualify the event.
Define how event 5 changes the state	EV5_CTRL	Set STATEV bits to 0 and the STATED bit to 1. Event 5 changes the state to state 0.
Define by which events output 0 is set	OUT0_SET	Set SET0 bits 0 (for event 0) and 3 (for event 3) to one to set the output when these events 0 and 3 occur.
Define by which events output 0 is cleared	OUT0_CLR	Set CLR0 bits 1 (for events 1) and 4 (for event 4) to one to clear the output when events 1 and 4 occur.
Configure states in which event 0 is enabled	EV0_STATE	Set STATEMSK0 bit 0 to 1. Set all other bits to 0. Event 0 is enabled in state 0.
Configure states in which event 1 is enabled	EV1_STATE	Set STATEMSK1 bit 0 to 1. Set all other bits to 0. Event 1 is enabled in state 0.
Configure states in which event 2 is enabled	EV2_STATE	Set STATEMSK2 bit 0 to 1. Set all other bits to 0. Event 2 is enabled in state 0.
Configure states in which event 3 is enabled	EV3_STATE	Set STATEMSK3 bit 1 to 1. Set all other bits to 0. Event 3 is enabled in state 1.
Configure states in which event 4 is enabled	EV4_STATE	Set STATEMSK4 bit 1 to 1. Set all other bits to 0. Event 4 is enabled in state 1.
Configure states in which event 5 is enabled	EV5_STATE	Set STATEMSK5 bit 1 to 1. Set all other bits to 0. Event 5 is enabled in state 1.

UM10732

Chapter 19: LPC11U6x/E6x 16-bit counter/timers CT16B0/1

Rev. 1.8 — 19 December 2016 **User manual**

19.1 How to read this chapter

The 16-bit counter/timers are available on all parts.

For LPC11E6x, the capture channel 1 is reserved for CT16B0.

19.2 Features

- Two 16-bit counter/timers with a programmable 16-bit prescaler.
- Counter or timer operation
- Two 16-bit capture channels that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- The timer and prescaler may be configured to be cleared on a designated capture event. This feature permits easy pulse-width measurement by clearing the timer on the leading edge of an input pulse and capturing the timer value on the trailing edge.
- Four 16-bit match registers that allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Two external outputs corresponding to match registers with the following capabilities:
  - Set LOW on match.
  - Set HIGH on match.
  - Toggle on match.
  - Do nothing on match.
- For each timer, up to four match registers can be configured as PWM allowing to use up to three match outputs as single-edge controlled PWM outputs.

19.3 Basic configuration

The CT16B0/1 counter/timers are configured through the following registers:

- Pins: The CT16B0/1 pins must be configured in the IOCON register block. See Table 83.
- Power: In the SYSAHBCLKCTRL register, set bit 7 and 8 in Table 40.
- The timer peripheral clock is determined by the system clock.
- Capture channel 1 of CT16B0 is connected to USB_FTOGGLE. See Section 15.4.6

19.4 General description

Each Counter/timer is designed to count cycles of the peripheral clock (PCLK) or an externally supplied clock and can optionally generate interrupts or perform other actions at specified timer values based on four match registers. Each counter/timer also includes one capture input to trap the timer value when an input signal transitions, optionally generating an interrupt.

In PWM mode, two match registers can be used to provide a single-edge controlled PWM output on the match output pins. It is recommended to use the match registers that are not pinned out to control the PWM cycle length.

19.4.1 Capture inputs

The capture signal can be configured to load the Capture Register with the value in the counter/timer and optionally generate an interrupt. The capture signal is either generated by one of the pins with a capture function or by the USB_FTOGGLE signal. Each capture signal is connected to one capture channel of the timer.

The Counter/Timer block can select a capture signal as a clock source instead of the PCLK derived clock. For more details see Section 19.6.11.

19.4.2 Match outputs

When a match register equals the timer counter (TC), the corresponding match output can either toggle, go LOW, go HIGH, or do nothing. The External Match Register (EMR) and the PWM Control Register (PWMCON) control the functionality of this output.

19.4.3 Block diagram

The block diagram for counter/timer0 and counter/timer1 is shown in Figure 71.

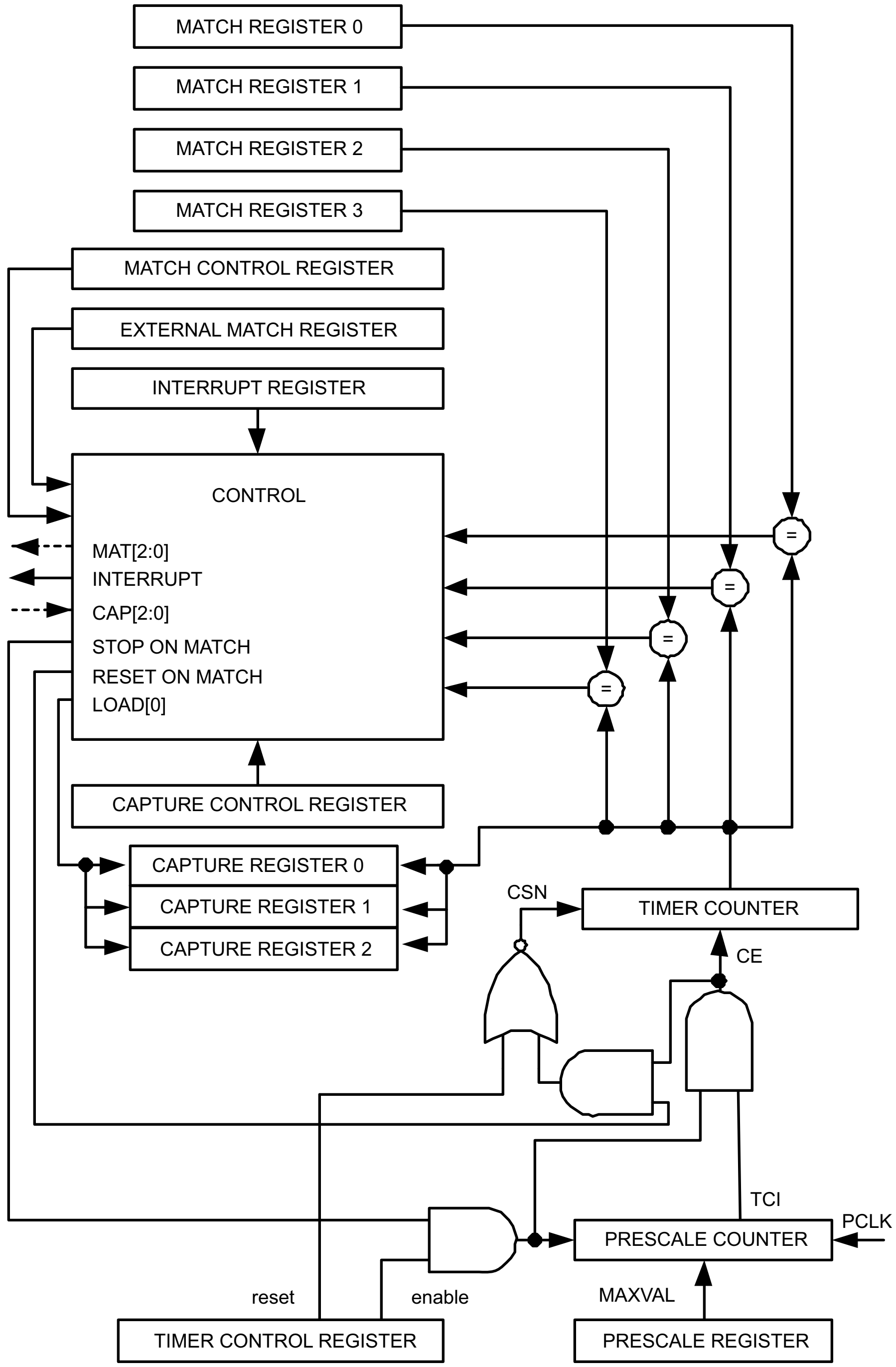


Fig 71. 16-bit counter/timer block diagram

19.4.4 Applications

- Interval timer for counting internal events
- Pulse Width Demodulator via capture input
- Free running timer

- Pulse Width Modulator via match outputs

19.5 Pin description

Table 297. Counter/timer pin description

Pin/signal	Type	Connected to	Description
CT16B0_CAP0	input from pin	CT16B0 channel 0	Pin CT160_CAP0 pin connected to capture channel 0 of CT16B0.
USB_FTOGGLE	internal	CT16B0 channel 1	USB_FTOGGLE signal generated by the USB block. This signal is connected to capture channel 1 of CT16B0.
CT16B0_CAP2	input from pin	CT16B0 channel 2	Pin CT16B0_CAP2 pin connected to capture channel 2 of CT16B_0.
CT16B1_CAP0	input from pin	CT16B1 channel 0	Pin CT161_CAP0 pin connected to capture channel 0 of CT16B1.
CT16B1_CAP1	input from pin	CT16B1 channel 1	Pin CT161_CAP1 pin connected to capture channel 1 of CT16B1.
CT16B0_MAT[2:0]	output to pin	CT16B0 channels 2 to 0	External match outputs of CT16B0
CT16B1_MAT[1:0]	output to pin	CT16B1 channels 1 to 0	External match outputs of CT16B1

19.6 Register description

Table 298. Register overview: 16-bit counter/timer 0 CT16B0 (base address 0x4000 C000)

Name	Access	Address offset	Description	Reset value[1]	Reference
IR	R/W	0x000	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	0	Table 300
TCR	R/W	0x004	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0	Table 301
TC	R/W	0x008	Timer Counter. The 16-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0	Table 302
PR	R/W	0x00C	Prescale Register. When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	0	Table 303
PC	R/W	0x010	Prescale Counter. The 16-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	0	Table 304
MCR	R/W	0x014	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	0	Table 305
MR0	R/W	0x018	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	0	Table 306
MR1	R/W	0x01C	Match Register 1. See MR0 description.	0	Table 306

Table 298. Register overview: 16-bit counter/timer 0 CT16B0 (base address 0x4000 C000) *…continued*

Name	Access	Address offset	Description	Reset value[1]	Reference
MR2	R/W	0x020	Match Register 2. See MR0 description.	0	Table 306
MR3	R/W	0x024	Match Register 3. See MR0 description.	0	Table 306
CCR	R/W	0x028	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	0	Table 307
CR0	RO	0x02C	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CT16B0_CAP0 input.	0	Table 308
CR1	RO	0x030	Capture register 1. CR1 is loaded with the value of TC when a USB_FTOGGLE signal occurs.	0	Table 308
CR2	RO	0x034	Capture Register 2. CR2 is loaded with the value of TC when there is an event on the CT16B0_CAP2 input.	0	Table 308
-	-	0x038	Reserved.	-	-
EMR	R/W	0x03C	External Match Register. The EMR controls the match function and the external match pins CT16B0_MAT[1:0] and CT16B1_MAT[1:0].	0	Table 309
-	-	0x040 - 0x06C	Reserved.	-	-
CTCR	R/W	0x070	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0	Table 311
PWMC	R/W	0x074	PWM Control Register. The PWMCON enables PWM mode for the external match pins CT16B0_MAT[1:0] and CT16B1_MAT[1:0].	0	Table 312

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

Table 299. Register overview: 16-bit counter/timer 1 CT16B1 (base address 0x4001 0000)

Name	Access	Address	Description	Reset value[1]	Reference
IR	R/W	0x000	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	0	Table 300
TCR	R/W	0x004	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0	Table 301
TC	R/W	0x008	Timer Counter. The 16-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0	Table 302
PR	R/W	0x00C	Prescale Register. When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	0	Table 303
PC	R/W	0x010	Prescale Counter. The 16-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	0	Table 304
MCR	R/W	0x014	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	0	Table 305

Table 299. Register overview: 16-bit counter/timer 1 CT16B1 (base address 0x4001 0000) *…continued*

Name	Access	Address	Description	Reset value[1]	Reference
MR0	R/W	0x018	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	0	Table 306
MR1	R/W	0x01C	Match Register 1. See MR0 description.	0	Table 306
MR2	R/W	0x020	Match Register 2. See MR0 description.	0	Table 306
MR3	R/W	0x024	Match Register 3. See MR0 description.	0	Table 306
CCR	R/W	0x028	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	0	Table 307
CR0	RO	0x02C	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CT16B1_CAP0 input.	0	Table 308
CR1	RO	0x030	Capture Register 1. CR1 is loaded with the value of TC when there is an event on the CT16B1_CAP1 input.	0	Table 308
-	-	0x034	Reserved.	-	-
-	-	0x038	Reserved.	-	-
EMR	R/W	0x03C	External Match Register. The EMR controls the match function and the external match pins CT16B0_MAT[2:0] and CT16B1_MAT[1:0].	0	Table 309
-	-	0x040 - 0x06C	Reserved.	-	-
CTCR	R/W	0x070	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0	Table 311
PWMC	R/W	0x074	PWM Control Register. The PWMCON enables PWM mode for the external match pins CT16B0_MAT[1:0] and CT16B1_MAT[1:0].	0	Table 312

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

19.6.1 Interrupt Register

The Interrupt Register consists of four bits for the match interrupts and two bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be HIGH. Otherwise, the bit will be LOW. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

Table 300. Interrupt Register (IR, address 0x4000 C000 (CT16B0) and 0x4001 0000 (CT16B1)) bit description

Bit	Symbol	Description	Reset value
0	MR0INT	Interrupt flag for match channel 0.	0
1	MR1INT	Interrupt flag for match channel 1.	0
2	MR2INT	Interrupt flag for match channel 2.	0
3	MR3INT	Interrupt flag for match channel 3.	0
4	CR0INT	Interrupt flag for capture channel 0 event.	0

Table 300. Interrupt Register (IR, address 0x4000 C000 (CT16B0) and 0x4001 0000 (CT16B1)) bit description

Bit	Symbol	Description	Reset value
5	CR1INT	Interrupt flag for capture channel 1 event.	-
6	CR2INT	Interrupt flag for capture channel 2 event.	0
31:7	-	Reserved	-

19.6.2 Timer Control Register

The Timer Control Register (TCR) is used to control the operation of the counter/timer.

Table 301. Timer Control Register (TCR, address 0x4000 C004 (CT16B0) and 0x4001 0004 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	CEN		Counter enable.	0
		0	The counters are disabled.	
		1	The Timer Counter and Prescale Counter are enabled for counting.	
1	CRST		Counter reset.	0
		0	Do nothing.	
		1	The Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero.	
31:2	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.6.3 Timer Counter

The 16-bit Timer Counter is incremented when the Prescale Counter reaches its terminal count. Unless it is reset before reaching its upper limit, the TC will count up to the value 0x0000 FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a Match register can be used to detect an overflow if needed.

Table 302: Timer counter registers (TC, address 0x4000 C008 (CT16B0) and 0x4001 0008 (CT16B1)) bit description

Bit	Symbol	Description	Reset value
15:0	TCVAL	Timer counter value.	0
31:16	-	Reserved.	-

19.6.4 Prescale Register

The 16-bit Prescale Register specifies the maximum value for the Prescale Counter.

Table 303: Prescale registers (PR, address 0x4000 C00C (CT16B0) and 0x4001 000C (CT16B1)) bit description

Bit	Symbol	Description	Reset value
15:0	PRVAL	Prescale value.	0
31:16	-	Reserved.	-

19.6.5 Prescale Counter register

The 16-bit Prescale Counter controls division of PCLK by some constant value before it is applied to the Timer Counter. This allows control of the relationship between the resolution of the timer and the maximum time before the timer overflows. The Prescale Counter is incremented on every PCLK. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented, and the Prescale Counter is reset on the next PCLK. This causes the TC to increment on every PCLK when PR = 0, every 2 PCLKs when PR = 1...

Table 304: Prescale counter registers (PC, address 0x4000 C010 (CT16B0) and 0x4001 0010 (CT16B1)) bit description

Bit	Symbol	Description	Reset value
15:0	PCVAL	Prescale counter value.	0
31:16	-	Reserved.	-

19.6.6 Match Control Register

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter. The function of each of the bits is shown in Table 305.

Table 305. Match Control Register (MCR, address 0x4000 C014 (CT16B0) and 0x4001 0014 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	MR0I		Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
1	MR0R		Reset on MR0: the TC will be reset if MR0 matches it.	0
		1	Enabled	
		0	Disabled	
2	MR0S		Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC.	0
		1	Enabled	
		0	Disabled	
3	MR1I		Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
4	MR1R		Reset on MR1: the TC will be reset if MR1 matches it.	0
		1	Enabled	
		0	Disabled	
5	MR1S		Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC.	0
		1	Enabled	
		0	Disabled	

Table 305. Match Control Register (MCR, address 0x4000 C014 (CT16B0) and 0x4001 0014 (CT16B1)) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
6	MR2I		Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
7	MR2R		Reset on MR2: the TC will be reset if MR2 matches it.	0
		1	Enabled	
		0	Disabled	
8	MR2S		Stop on MR2: the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC.	0
		1	Enabled	
		0	Disabled	
9	MR3I		Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
10	MR3R		Reset on MR3: the TC will be reset if MR3 matches it.	0
		1	Enabled	
		0	Disabled	
11	MR3S		Stop on MR3: the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC.	0
		1	Enabled	
		0	Disabled	
31:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.6.7 Match Registers

The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

Table 306: Match registers (MR[0:3], addresses 0x4000 C018 (MR0) to 0x4000 C024 (MR3) (CT16B0) and 0x4001 0018 (MR0) to 0x4001 0024 (MR3) (CT16B1)) bit description

Bit	Symbol	Description	Reset value
15:0	MATCH	Timer counter match value.	0
31:16	-	Reserved.	-

19.6.8 Capture Control Register

The Capture Control Register is used to control whether the Capture Register is loaded with the value in the Counter/timer when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges.

Table 307. Capture Control Register (CCR, address 0x4000 C028 (CT16B0) and 0x4001 0028 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	CAP0RE		Rising edge of capture channel 0: a sequence of 0 then 1 causes CR0 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
1	CAP0FE		Falling edge of capture channel 0: a sequence of 1 then 0 causes CR0 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
2	CAP0I		Generate interrupt on channel 0 capture event: a CR0 load generates an interrupt.	0
		1	Enabled.	
		0	Disabled.	
3	CAP1RE		Rising edge of capture channel 1: a sequence of 0 then 1 causes CR1 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
4	CAP1FE		Falling edge of capture channel 1: a sequence of 1 then 0 causes CR1 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
5	CAP1I		Generate interrupt on channel 1 capture event: a CR1 load generates an interrupt.	0
		1	Enabled.	
		0	Disabled.	
6	CAP2RE		Rising edge of capture channel 2: a sequence of 0 then 1 causes CR2 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
7	CAP2FE		Falling edge of capture channel 2: a sequence of 1 then 0 causes CR2 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
8	CAP2I		Generate interrupt on channel 2 capture event: a CR2 load generates an interrupt.	0
		1	Enabled.	
		0	Disabled.	
31:9	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.6.9 Capture Registers

Each Capture register is associated with one capture channel and may be loaded with the counter/timer value when a specified event occurs on the signal defined for that capture channel. The signal can originate from an external pin or from an internal source. The settings in the Capture Control Register register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated signal, the falling edge, or on both edges.

Table 308: Capture registers (CR[0:2], address 0x4000 C02C(CR0) to 0x4000 C034 (CR2) (CT16B0) and address 0x4001 002C(CR0) to 0x4001 0030 (CR1) (CT16B1)) bit description

Bit	Symbol	Description	Reset value
15:0	CAP	Timer counter capture value.	0
31:16	-	Reserved.	-

19.6.10 External Match Register

The External Match Register provides both control and status of the external match pins CT16Bn_MAT[1:0].

If the match outputs are configured as PWM output, the function of the external match registers is determined by the PWM rules (Section 19.7.1 “Rules for single edge controlled PWM outputs” on page 382).

Table 309. External Match Register (EMR, address 0x4000 C03C (CT16B0) and 0x4001 003C (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	EM0		External Match 0. This bit reflects the state of output CT16B0_MAT0/CT16B1_MAT0, whether or not this output is connected to its pin. When a match occurs between the TC and MR0, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[5:4] control the functionality of this output. This bit is driven to the CT16B0_MAT0/CT16B1_MAT0 pins if the match function is selected in the IOCON registers (0 = LOW, 1 = HIGH).	0
1	EM1		External Match 1. This bit reflects the state of output CT16B0_MAT1/CT16B1_MAT1, whether or not this output is connected to its pin. When a match occurs between the TC and MR1, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[7:6] control the functionality of this output. This bit is driven to the CT16B0_MAT0/CT16B1_MAT0 pins if the match function is selected in the IOCON registers (0 = LOW, 1 = HIGH).	0
2	EM2		External Match 2. This bit reflects the state of match channel 2. When a match occurs between the TC and MR2, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[9:8] control the functionality of this output.	0
3	EM3		External Match 3. This bit reflects the state of output of match channel 3. When a match occurs between the TC and MR3, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[11:10] control the functionality of this output.	0
5:4	EMC0		External Match Control 0. Determines the functionality of External Match 0. Table 310 shows the encoding of these bits.	00
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (CT16Bn_MAT0 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (CT16Bn_MAT0 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	

Table 309. External Match Register (EMR, address 0x4000 C03C (CT16B0) and 0x4001 003C (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
7:6	EMC1		External Match Control 1. Determines the functionality of External Match 1.	00
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (CT16Bn_MAT1 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (CT16Bn_MAT1 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
9:8	EMC2		External Match Control 2. Determines the functionality of External Match 2.	00
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (CT16Bn_MAT2 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (CT16Bn_MAT2 pin is HIGH if pinned out).	
		0x3	Toggle.Toggle the corresponding External Match bit/output.	
11:10	EMC3		External Match Control 3. Determines the functionality of External Match 3.	00
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (CT16Bn_MAT3 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (CT16Bn_MAT3 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
31:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 310. External match control

EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	Function
00	Do Nothing.
01	Clear the corresponding External Match bit/output to 0 (CT16Bn_MATm pin is LOW if pinned out).
10	Set the corresponding External Match bit/output to 1 (CT16Bn_MATm pin is HIGH if pinned out).
11	Toggle the corresponding External Match bit/output.

19.6.11 Count Control Register

The Count Control Register (CTCR) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edges for counting.

When Counter Mode is chosen as a mode of operation, the CAP input (selected by the CTCR bits 3:2) is sampled on every rising edge of the PCLK clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected CAP input. Only if the identified event occurs, and the event corresponds to the one selected by bits 1:0 in the CTCR register, will the Timer Counter register be incremented.

Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the PCLK clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input cannot exceed one half of the PCLK clock. Consequently, the duration of the HIGH/LOW levels on the same CAP input in this case can not be shorter than 1/PCLK.

Bits 7:4 of this register are also used to enable and configure the capture-clears-timer feature. This feature allows for a designated edge on a particular CAP input to reset the timer to all zeros. Using this mechanism to clear the timer on the leading edge of an input pulse and performing a capture on the trailing edge, permits direct pulse-width measurement using a single capture input without the need to perform a subtraction operation in software.

Table 311. Count Control Register (CTCR, address 0x4000 C070 (CT16B0) and 0x4001 0070 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
1:0	CTM		Counter/Timer Mode. This field selects which rising PCLK edges can increment Timer’s Prescale Counter (PC), or clear PC and increment Timer Counter (TC). **Remark:** If Counter mode is selected in the CTCR, bits 2:0 in the Capture Control Register (CCR) must be programmed as 000.	0
		0x0	Timer Mode. Increments every rising PCLK edge	
		0x1	Counter Mode rising edge. TC is incremented on rising edges on the CAP input selected by bits 3:2.	
		0x2	Counter Mode falling edge: TC is incremented on falling edges on the CAP input selected by bits 3:2.	
		0x3	Counter Mode dual edge: TC is incremented on both edges on the CAP input selected by bits 3:2.	
3:2	CIS		Count Input Select. In counter mode (when bits 1:0 in this register are not 00), these bits select which CAP pin is sampled for clocking. Value 0x3 is reserved.	0
		0x0	Capture channel 0.	
		0x1	Capture channel 1.	
		0x2	Capture channel 2.	
4	ENCC		Setting this bit to 1 enables clearing of the timer and the prescaler when the capture-edge event specified in bits 7:5 occurs.	0

Table 311. Count Control Register (CTCR, address 0x4000 C070 (CT16B0) and 0x4001 0070 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
7:5	SELCC		Edge select. When bit 4 is 1, these bits select which capture input edge will cause the timer and prescaler to be cleared. These bits have no effect when bit 4 is low. Values 0x2 to 0x3 and 0x6 to 0x7 are reserved.	0
		0x0	Rising Edge of the signal on capture channel 0 clears the timer (if bit 4 is set).	
		0x1	Falling Edge of the signal on capture channel 0 clears the timer (if bit 4 is set).	
		0x2	Rising Edge of the signal on capture channel 1 clears the timer (if bit 4 is set).	
		0x3	Falling Edge of the signal on capture channel 1 clears the timer (if bit 4 is set).	
		0x4	Rising Edge of the signal on capture channel 2 clears the timer (if bit 4 is set).	
		0x5	Falling Edge of the signal on capture channel 2 clears the timer (if bit 4 is set).	
31:8	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

19.6.12 PWM Control register

The PWM Control Register is used to configure the match outputs as PWM outputs. Each match output can be independently set to perform either as PWM output or as match output whose function is controlled by the External Match Register (EMR).

For each timer, a maximum of three single edge controlled PWM outputs can be selected on the CT16Bn_MAT[1:0] outputs. One additional match register determines the PWM cycle length. When a match occurs in any of the other match registers, the PWM output is set to HIGH. The timer is reset by the match register that is configured to set the PWM cycle length. When the timer is reset to zero, all currently HIGH match outputs configured as PWM outputs are cleared.

Table 312. PWM Control Register (PWMC, address 0x4000 C074 (CT16B0) and 0x4001 0074 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	PWMEN0		PWM mode enable for channel0.	0
		0	CT16Bn_MAT0 is controlled by EM0.	
		1	PWM mode is enabled for CT16Bn_MAT0.	
1	PWMEN1		PWM mode enable for channel1.	0
		0	CT16Bn_MAT01 is controlled by EM1.	
		1	PWM mode is enabled for CT16Bn_MAT1.	
2	PWMEN2		PWM mode enable for channel2.	0
		0	CT16Bn_MAT2 is controlled by EM2.	
		1	PWM mode is enabled for CT16Bn_MAT2.	

Table 312. PWM Control Register (PWMC, address 0x4000 C074 (CT16B0) and 0x4001 0074 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
3	PWMEN3		PWM mode enable for channel3.	0
		0	CT16Bn_MAT3 is controlled by EM3.	
		1	PWM mode is enabled for CT16Bn_MAT3.	
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

19.7 Functional description

Figure 72 shows a timer configured to reset the count and generate an interrupt on match. The prescaler is set to 2 and the match register set to 6. At the end of the timer cycle where the match occurs, the timer count is reset. This gives a full length cycle to the match value. The interrupt indicating that a match occurred is generated in the next clock after the timer reached the match value.

Figure 73 shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

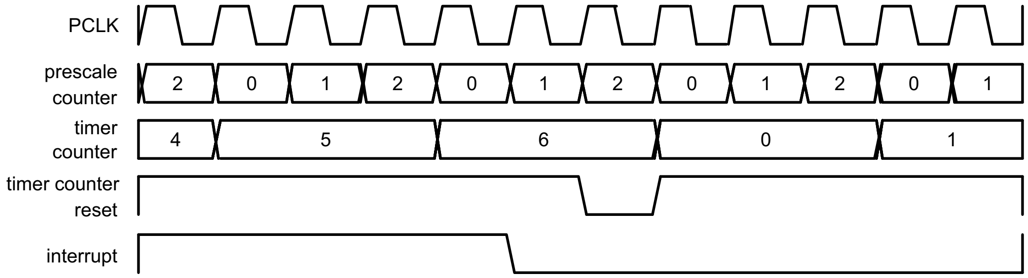


Fig 72. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

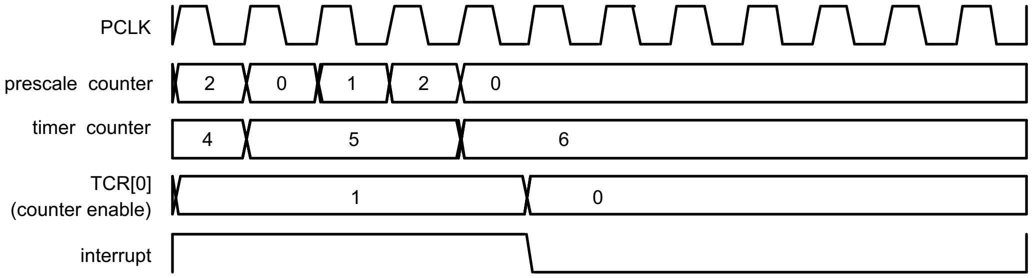


Fig 73. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

19.7.1 Rules for single edge controlled PWM outputs

1. All single edge controlled PWM outputs go LOW at the beginning of each PWM cycle (timer is set to zero) unless their match value is equal to zero.
2. Each PWM output will go HIGH when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM cycle length), the PWM output remains continuously LOW.
3. If a match value larger than the PWM cycle length is written to the match register, and the PWM signal is HIGH already, then the PWM signal will be cleared on the next start of the next PWM cycle.
4. If a match register contains the same value as the timer reset value (the PWM cycle length), then the PWM output will be reset to LOW on the next clock tick. Therefore, the PWM output will always consist of a one clock tick wide positive pulse with a period determined by the PWM cycle length (i.e. the timer reload value).
5. If a match register is set to zero, then the PWM output will go to HIGH the first time the timer goes back to zero and will stay HIGH continuously.

Note: When the match outputs are selected to perform as PWM outputs, the timer reset (MRnR) and timer stop (MRnS) bits in the Match Control Register MCR must be set to zero except for the match register setting the PWM cycle length. For this register, set the MRnR bit to one to enable the timer reset when the timer value matches the value of the corresponding match register.

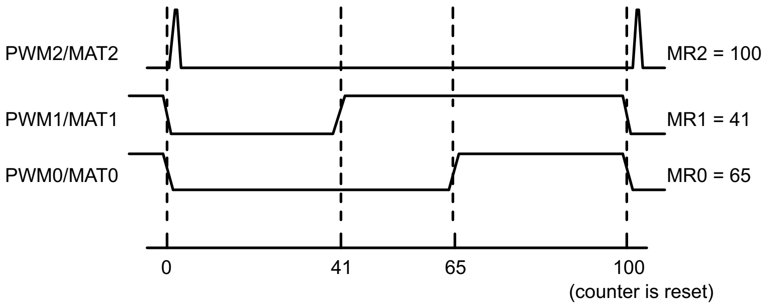


Fig 74. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR2) and MAT2:0 enabled as PWM outputs by the PWMC register.

UM10732

Chapter 20: LPC11U6x/E6x 32-bit counter/timers CT32B0/1

Rev. 1.8 — 19 December 2016 **User manual**

20.1 How to read this chapter

The 32-bit counter/timers are available on all parts.

For LPC11E6x, the capture channel 1 is reserved for CT32B0.

20.2 Basic configuration

The CT32B0/1 counter/timers are configured through the following registers:

- Pins: The CT32B0/1 pins must be configured in the IOCON register block. See Table 83.
- Power: In the SYSAHBCLKCTRL register, set bit 9 and 10 in Table 40.
- The peripheral clock is determined by the system clock.
- Capture channel 1 of CT32B0 is connected to USB_FTOGGLE. See Section 15.4.6.

20.3 Features

- Two 32-bit counter/timers with a programmable 32-bit prescaler.
- Counter or timer operation.
- Four 32-bit capture channels that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- The timer and prescaler may be configured to be cleared on a designated capture event. This feature permits easy pulse-width measurement by clearing the timer on the leading edge of an input pulse and capturing the timer value on the trailing edge.
- Four 32-bit match registers that allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Four external outputs corresponding to match registers with the following capabilities:
  - Set LOW on match.
  - Set HIGH on match.
  - Toggle on match.
  - Do nothing on match.
- For each timer, up to four match registers can be configured as PWM allowing to use up to three match outputs as single edge controlled PWM outputs.

20.4 General description

Each Counter/timer is designed to count cycles of the peripheral clock (PCLK) or an externally supplied clock and can optionally generate interrupts or perform other actions at specified timer values based on four match registers. Each counter/timer also includes one capture input to trap the timer value when an input signal transitions, optionally generating an interrupt.

In PWM mode, three match registers can be used to provide a single-edge controlled PWM output on the match output pins. One match register is used to control the PWM cycle length.

20.4.1 Capture inputs

The capture signal can be configured to load the Capture Register with the value in the counter/timer and optionally generate an interrupt. The capture signal is either generated by one of the pins with a capture function or by the USB_FTOGGLE signal. Each capture signal is connected to one capture channel of the timer.

The Counter/Timer block can select a capture signal as a clock source instead of the PCLK derived clock. For more details see Section 20.6.11.

20.4.2 Match outputs

When a match register equals the timer counter (TC), the corresponding match output can either toggle, go LOW, go HIGH, or do nothing. The External Match Register (EMR) and the PWM Control Register (PWMCON) control the functionality of this output.

20.4.3 Architecture

The block diagram for 32-bit counter/timer0 and 32-bit counter/timer1 is shown in Figure 75.

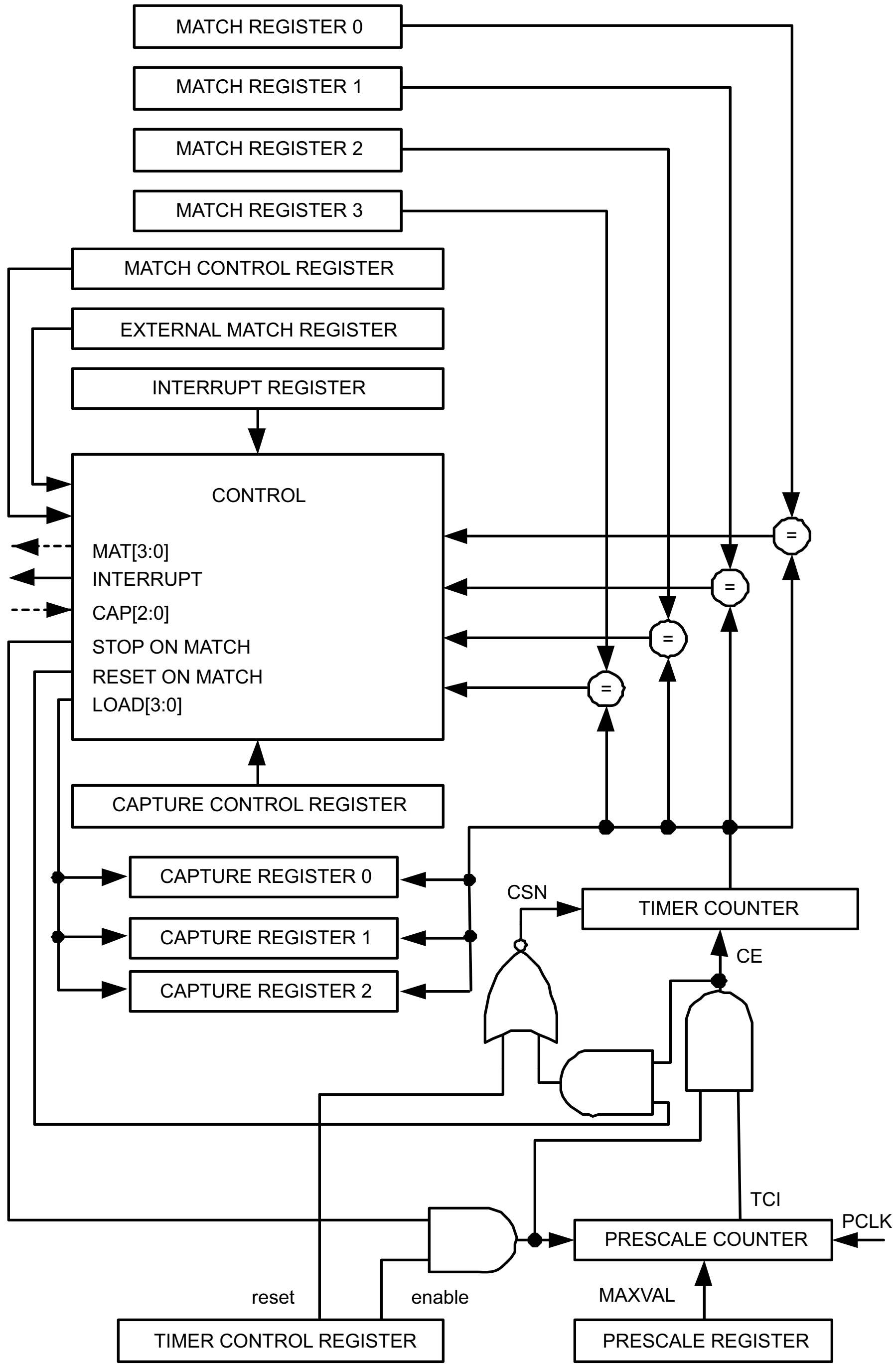


Fig 75. 32-bit counter/timer block diagram

20.4.4 Applications

- Interval timer for counting internal events
- Pulse Width Demodulator via capture input
- Free running timer

- Pulse Width Modulator via match outputs

20.5 Pin description

.

Table 313. Counter/timer pin description

Pin/signal	Type	Connected to	Description
CT32B0_CAP0	input from pin	CT32B0 channel 0	Pin CT32B_CAP0 pin connected to capture channel 0 of CT32B0.
USB_FTOGGLE	internal	CT32B0 channel 1	USB_FTOGGLE signal generated by the USB block. This signal is connected to capture channel 1 of CT32B0.
CT32B0_CAP2	input from pin	CT32B0 channel 2	Pin CT32B0_CAP2 pin connected to capture channel 2 of CT32B_0.
CT32B1_CAP0	input from pin	CT32B1 channel 0	Pin CT32B1_CAP0 pin connected to capture channel 0 of CT32B1.
CT32B1_CAP1	input from pin	CT32B1 channel 1	Pin CT32B1_CAP1 pin connected to capture channel 1 of CT32B1.
CT32B0_MAT[3:0]	output to pin	CT32B0 channels 3 to 0	External match outputs of CT32B0
CT32B1_MAT[3:0]	output to pin	CT32B1 channels 3 to 0	External match outputs of CT32B1

20.6 Register description

Table 314. Register overview: 32-bit counter/timer 0 CT32B0 (base address 0x4001 4000)

Name	Access	Address offset	Description	Reset value[1]	Reference
IR	R/W	0x000	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	0	Table 316
TCR	R/W	0x004	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0	Table 317
TC	R/W	0x008	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0	Table 318
PR	R/W	0x00C	Prescale Register. When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	0	Table 319
PC	R/W	0x010	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	0	Table 320
MCR	R/W	0x014	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	0	Table 321
MR0	R/W	0x018	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	0	Table 322

Table 314. Register overview: 32-bit counter/timer 0 CT32B0 (base address 0x4001 4000) *…continued*

Name	Access	Address offset	Description	Reset value[1]	Reference
MR1	R/W	0x01C	Match Register 1. See MR0 description.	0	Table 322
MR2	R/W	0x020	Match Register 2. See MR0 description.	0	Table 322
MR3	R/W	0x024	Match Register 3. See MR0 description.	0	Table 322
CCR	R/W	0x028	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	0	Table 323
CR0	RO	0x02C	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CT32B0_CAP0 input.	0	Table 324
CR1	-	0x030	Capture register 1. CR1 is loaded with the value of TC based on the USB_FTOGGLE signal.	-	Table 324
CR2	-	0x034	Capture Register 2. CR2 is loaded with the value of TC when there is an event on the CT32B0_CAP2 input.	-	Table 324
-	-	0x038	Reserved.	-	-
EMR	R/W	0x03C	External Match Register. The EMR controls the match function and the external match pins CT32Bn_MAT[3:0].	0	Table 325
-	-	0x040 - 0x06C	Reserved.	-	-
CTCR	R/W	0x070	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0	Table 327
PWMC	R/W	0x074	PWM Control Register. The PWMCON enables PWM mode for the external match pins CT32Bn_MAT[3:0].	0	Table 328

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

Table 315. Register overview: 32-bit counter/timer 1 CT32B1 (base address 0x4001 8000)

Name	Access	Address offset	Description	Reset value[1]	Reference
IR	R/W	0x000	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	0	Table 316
TCR	R/W	0x004	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0	Table 317
TC	R/W	0x008	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0	Table 318
PR	R/W	0x00C	Prescale Register. When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	0	Table 319
PC	R/W	0x010	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	0	Table 320

Table 315. Register overview: 32-bit counter/timer 1 CT32B1 (base address 0x4001 8000) *…continued*

Name	Access	Address offset	Description	Reset value[1]	Reference
MCR	R/W	0x014	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	0	Table 321
MR0	R/W	0x018	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	0	Table 322
MR1	R/W	0x01C	Match Register 1. See MR0 description.	0	Table 322
MR2	R/W	0x020	Match Register 2. See MR0 description.	0	Table 322
MR3	R/W	0x024	Match Register 3. See MR0 description.	0	Table 322
CCR	R/W	0x028	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	0	Table 323
CR0	RO	0x02C	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CT32B1_CAP0 input.	0	Table 324
CR1	RO	0x030	Capture Register 1. CR1 is loaded with the value of TC when there is an event on the CT32B1_CAP1 input.	0	Table 324
-	-	0x034	Reserved.	-	-
-	-	0x038	Reserved.	-	-
EMR	R/W	0x03C	External Match Register. The EMR controls the match function and the external match pins CT32Bn_MAT[3:0].	0	Table 325
-	-	0x040 - 0x06C	Reserved.	-	-
CTCR	R/W	0x070	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0	Table 327
PWMC	R/W	0x074	PWM Control Register. The PWMCON enables PWM mode for the external match pins CT32Bn_MAT[3:0].	0	Table 328

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

20.6.1 Interrupt Register

The Interrupt Register consists of four bits for the match interrupts and four bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be HIGH. Otherwise, the bit will be LOW. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

Table 316: Interrupt Register (IR, address 0x4001 4000 (CT32B0) and 0x4001 8000 (CT32B1)) bit description

Bit	Symbol	Description	Reset value
0	MR0INT	Interrupt flag for match channel 0.	0
1	MR1INT	Interrupt flag for match channel 1.	0
2	MR2INT	Interrupt flag for match channel 2.	0
3	MR3INT	Interrupt flag for match channel 3.	0

Table 316: Interrupt Register (IR, address 0x4001 4000 (CT32B0) and 0x4001 8000 (CT32B1)) bit description

Bit	Symbol	Description	Reset value
4	CR0INT	Interrupt flag for capture channel 0 event.	0
5	CR1INT	Interrupt flag for capture channel 1 event.	0
6	CR2INT	Interrupt flag for capture channel 2 event.	0
31:7	-	Reserved	-

20.6.2 Timer Control Register

The Timer Control Register (TCR) is used to control the operation of the counter/timer.

Table 317: Timer Control Register (TCR, address 0x4001 4004 (CT32B0) and 0x4001 8004 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	CEN		Counter enable.	0
		0	The counters are disabled.	
		1	The Timer Counter and Prescale Counter are enabled for counting.	
1	CRST		Counter reset.	0
		0	Do nothing.	
		1	The Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero.	
31:2	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

20.6.3 Timer Counter registers

The 32-bit Timer Counter is incremented when the Prescale Counter reaches its terminal count. Unless it is reset before reaching its upper limit, the TC will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a Match register can be used to detect an overflow if needed.

Table 318: Timer counter registers (TC, address 0x4001 4008 (CT32B0) and 0x4001 8008 (CT32B1)) bit description

Bit	Symbol	Description	Reset value
31:0	TCVAL	Timer counter value.	0

20.6.4 Prescale Register

The 32-bit Prescale Register specifies the maximum value for the Prescale Counter.

Table 319: Prescale registers (PR, address 0x4001 400C (CT32B0) and 0x4001 800C (CT32B1)) bit description

Bit	Symbol	Description	Reset value
31:0	PRVAL	Prescaler value.	0

20.6.5 Prescale Counter Register

The 32-bit Prescale Counter controls division of PCLK by some constant value before it is applied to the Timer Counter. This allows control of the relationship between the resolution of the timer and the maximum time before the timer overflows. The Prescale Counter is incremented on every PCLK. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented, and the Prescale Counter is reset on the next PCLK. This causes the TC to increment on every PCLK when PR = 0, every 2 PCLKs when PR = 1, etc.

Table 320: Prescale registers (PC, address 0x4001 4010 (CT32B0) and 0x4001 8010 (CT32B1)) bit description

Bit	Symbol	Description	Reset value
31:0	PCVAL	Prescale counter value.	0

20.6.6 Match Control Register

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter. The function of each of the bits is shown in Table 321.

Table 321: Match Control Register (MCR, address 0x4001 4014 (CT32B0) and 0x4001 8014 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	MR0I		Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
1	MR0R		Reset on MR0: the TC will be reset if MR0 matches it.	0
		1	Enabled	
		0	Disabled	
2	MR0S		Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC.	0
		1	Enabled	
		0	Disabled	
3	MR1I		Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
4	MR1R		Reset on MR1: the TC will be reset if MR1 matches it.	0
		1	Enabled	
		0	Disabled	

Table 321: Match Control Register (MCR, address 0x4001 4014 (CT32B0) and 0x4001 8014 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
5	MR1S		Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC.	0
		1	Enabled	
		0	Disabled	
6	MR2I		Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
7	MR2R		Reset on MR2: the TC will be reset if MR2 matches it.	0
		1	Enabled	
		0	Disabled	
8	MR2S		Stop on MR2: the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC.	0
		1	Enabled	
		0	Disabled	
9	MR3I		Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
10	MR3R		Reset on MR3: the TC will be reset if MR3 matches it.	0
		1	Enabled	
		0	Disabled	
11	MR3S		Stop on MR3: the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC.	0
		1	Enabled	
		0	Disabled	
31:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

20.6.7 Match Registers

The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

Table 322: Match registers (MR[0:3], addresses 0x4001 4018 (MR0) to 0x4001 4024 (MR3) (CT32B0) and 0x4001 8018(MR0) to 0x40018024 (MR3) (CT32B1)) bit description

Bit	Symbol	Description	Reset value
31:0	MATCH	Timer counter match value.	0

20.6.8 Capture Control Register

The Capture Control Register is used to control whether the Capture Register is loaded with the value in the Counter/timer when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges.

Table 323. Capture Control Register (CCR, address 0x4001 4028 (CT32B0) and 0x4001 8028 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	CAP0RE		Rising edge of capture channel 0: a sequence of 0 then 1 causes CR0 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
1	CAP0FE		Falling edge of capture channel 0: a sequence of 1 then 0 causes CR0 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
2	CAP0I		Generate interrupt on channel 0 capture event: a CR0 load generates an interrupt.	0
		1	Enabled.	
		0	Disabled.	
3	CAP1RE		Rising edge of capture channel 1: a sequence of 0 then 1 causes CR1 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
4	CAP1FE		Falling edge of capture channel 1: a sequence of 1 then 0 causes CR1 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
5	CAP1I		Generate interrupt on channel 1 capture event: a CR1 load generates an interrupt.	0
		1	Enabled.	
		0	Disabled.	
3	CAP2RE		Rising edge of capture channel 2: a sequence of 0 then 1 causes CR2 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
4	CAP2FE		Falling edge of capture channel 2: a sequence of 1 then 0 causes CR2 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
5	CAP2I		Generate interrupt on channel 2 capture event: a CR2 load generates an interrupt.	0
		1	Enabled.	
		0	Disabled.	
31:6	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

20.6.9 Capture Registers

Each Capture register is associated with one capture channel and may be loaded with the counter/timer value when a specified event occurs on the signal defined for that capture channel. The signal can originate from an external pin or from an internal source. The settings in the Capture Control Register register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated signal, the falling edge, or on both edges.

Table 324: Capture registers (CR[0:2], address 0x4001 402C (CR0) to 0x4001 4034 (CR2) (CT16B0) and address 0x4001 802C(CR0) to 0x4001 8030 (CR1) (CT16B1)) bit description

Bit	Symbol	Description	Reset value
31:0	CAP	Timer counter capture value.	0

20.6.10 External Match Register

The External Match Register provides both control and status of the external match pins CAP32Bn_MAT[3:0].

If the match outputs are configured as PWM output, the function of the external match registers is determined by the PWM rules (Section 20.7.1 “Rules for single edge controlled PWM outputs” on page 398).

Table 325: External Match Register (EMR, address 0x4001 403C (CT32B0) and 0x4001 803C (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	EM0		External Match 0. This bit reflects the state of output CT32Bn_MAT0, whether or not this output is connected to its pin. When a match occurs between the TC and MR0, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[5:4] control the functionality of this output. This bit is driven to the CT32B0_MAT0/CT32B1_MAT0 pins if the match function is selected in the IOCON registers (0 = LOW, 1 = HIGH).	0
1	EM1		External Match 1. This bit reflects the state of output CT32Bn_MAT1, whether or not this output is connected to its pin. When a match occurs between the TC and MR1, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[7:6] control the functionality of this output. This bit is driven to the CT32B0_MAT1/CT32B1_MAT1 pins if the match function is selected in the IOCON registers (0 = LOW, 1 = HIGH).	0
2	EM2		External Match 2. This bit reflects the state of output CT32Bn_MAT2, whether or not this output is connected to its pin. When a match occurs between the TC and MR2, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[9:8] control the functionality of this output. This bit is driven to the CT32B0_MAT2/CT32B1_MAT2 pins if the match function is selected in the IOCON registers (0 = LOW, 1 = HIGH).	0
3	EM3		External Match 3. This bit reflects the state of output CT32Bn_MAT3, whether or not this output is connected to its pin. When a match occurs between the TC and MR3, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[11:10] control the functionality of this output. This bit is driven to the CT32B3_MAT0/CT32B1_MAT3 pins if the match function is selected in the IOCON registers (0 = LOW, 1 = HIGH).	0

Table 325: External Match Register (EMR, address 0x4001 403C (CT32B0) and 0x4001 803C (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
5:4	EMC0		External Match Control 0. Determines the functionality of External Match 0.	00
		0x0	Do Nothing.	
		0x1	Clear the corresponding External Match bit/output to 0 (CT32Bi_MAT0 pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT32Bi_MAT0 pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	
7:6	EMC1		External Match Control 1. Determines the functionality of External Match 1.	00
		0x0	Do Nothing.	
		0x1	Clear the corresponding External Match bit/output to 0 (CT32Bi_MAT1 pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT32Bi_MAT1 pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	
9:8	EMC2		External Match Control 2. Determines the functionality of External Match 2.	00
		0x0	Do Nothing.	
		0x1	Clear the corresponding External Match bit/output to 0 (CT32Bi_MAT2 pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT32Bi_MAT2 pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	
11:10	EMC3		External Match Control 3. Determines the functionality of External Match 3.	00
		0x0	Do Nothing.	
		0x1	Clear the corresponding External Match bit/output to 0 (CT32Bi_MAT3 pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT32Bi_MAT3 pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	
31:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 326. External match control

EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	Function
00	Do Nothing.
01	Clear the corresponding External Match bit/output to 0 (CT32Bn_MATm pin is LOW if pinned out).
10	Set the corresponding External Match bit/output to 1 (CT32Bn_MATm pin is HIGH if pinned out).
11	Toggle the corresponding External Match bit/output.

20.6.11 Count Control Register

The Count Control Register (CTCR) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edges for counting.

When Counter Mode is chosen as a mode of operation, the CAP input (selected by the CTCR bits 3:2) is sampled on every rising edge of the PCLK clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected CAP input. Only if the identified event occurs, and the event corresponds to the one selected by bits 1:0 in the CTCR register, will the Timer Counter register be incremented.

Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the PCLK clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input cannot exceed one half of the PCLK clock. Consequently, duration of the HIGH/LOWLOW levels on the same CAP input in this case cannot be shorter than 1/PCLK.

Bits 7:4 of this register are also used to enable and configure the capture-clears-timer feature. This feature allows for a designated edge on a particular CAP input to reset the timer to all zeros. Using this mechanism to clear the timer on the leading edge of an input pulse and performing a capture on the trailing edge, permits direct pulse-width measurement using a single capture input without the need to perform a subtraction operation in software.

Table 327. Count Control Register (CTCR, address 0x4001 4070 (CT32B0) and 0x4001 8070 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
1:0	CTM		Counter/Timer Mode. This field selects which rising PCLK edges can increment Timer's Prescale Counter (PC), or clear PC and increment Timer Counter (TC). **Remark:** If Counter mode is selected in the CTCR, bits 2:0 in the Capture Control Register (CCR) must be programmed as 000.	0
		0x0	Timer Mode. Increments every rising PCLK edge	
		0x1	Counter Mode rising edge. TC is incremented on rising edges on the CAP input selected by bits 3:2.	
		0x2	Counter Mode falling edge: TC is incremented on falling edges on the CAP input selected by bits 3:2.	
		0x3	Counter Mode dual edge: TC is incremented on both edges on the CAP input selected by bits 3:2.	
3:2	CIS		Count Input Select. In counter mode (when bits 1:0 in this register are not 00), these bits select which CAP pin is sampled for clocking. Value 0x3 is reserved.	0
		0x0	Capture channel 0.	
		0x1	Capture channel 1.	
		0x2	Capture channel 2.	
4	ENCC		Setting this bit to 1 enables clearing of the timer and the prescaler when the capture-edge event specified in bits 7:5 occurs.	0

Table 327. Count Control Register (CTCR, address 0x4001 4070 (CT32B0) and 0x4001 8070 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
7:5	SELCC		Edge select. When bit 4 is 1, these bits select which capture input edge will cause the timer and prescaler to be cleared. These bits have no effect when bit 4 is low. Values 0x2 to 0x3 and 0x6 to 0x7 are reserved.	0
		0x0	Rising Edge of the signal on capture channel 0 clears the timer (if bit 4 is set).	
		0x1	Falling Edge of the signal on capture channel 0 clears the timer (if bit 4 is set).	
		0x2	Rising Edge of the signal on capture channel 1 clears the timer (if bit 4 is set).	
		0x3	Falling Edge of the signal on capture channel 1 clears the timer (if bit 4 is set).	
		0x4	Rising Edge of the signal on capture channel 2 clears the timer (if bit 4 is set).	
		0x5	Falling Edge of the signal on capture channel 2 clears the timer (if bit 4 is set).	
31:8	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

20.6.12 PWM Control Register

The PWM Control Register is used to configure the match outputs as PWM outputs. Each match output can be independently set to perform either as PWM output or as match output whose function is controlled by the External Match Register (EMR).

For each timer, a maximum of three single edge controlled PWM outputs can be selected on the MATn.2:0 outputs. One additional match register determines the PWM cycle length. When a match occurs in any of the other match registers, the PWM output is set to HIGH. The timer is reset by the match register that is configured to set the PWM cycle length. When the timer is reset to zero, all currently HIGH match outputs configured as PWM outputs are cleared.

Table 328: PWM Control Register (PWMC, 0x4001 4074 (CT32B0) and 0x4001 8074 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	PWMEN0		PWM mode enable for channel0.	0
		0	CT32Bn_MAT0 is controlled by EM0.	
		1	PWM mode is enabled for CT32Bn_MAT0.	
1	PWMEN1		PWM mode enable for channel1.	0
		0	CT32Bn_MAT01 is controlled by EM1.	
		1	PWM mode is enabled for CT32Bn_MAT1.	
2	PWMEN2		PWM mode enable for channel2.	0
		0	CT32Bn_MAT2 is controlled by EM2.	
		1	PWM mode is enabled for CT32Bn_MAT2.	

Table 328: PWM Control Register (PWMC, 0x4001 4074 (CT32B0) and 0x4001 8074 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
3	PWMEN3		PWM mode enable for channel3. **Note:** It is recommended to use match channel 3 to set the PWM cycle.	0
		0	CT32Bn_MAT3 is controlled by EM3.	
		1	PWM mode is enabled for CT132Bn_MAT3.	
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

20.7 Functional description

Figure 76 shows a timer configured to reset the count and generate an interrupt on match. The prescaler is set to 2 and the match register set to 6. At the end of the timer cycle where the match occurs, the timer count is reset. This gives a full length cycle to the match value. The interrupt indicating that a match occurred is generated in the next clock after the timer reached the match value.

Figure 77 shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

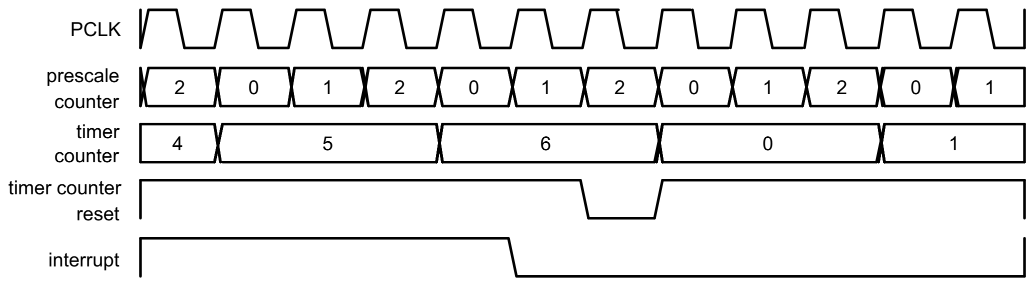


Fig 76. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

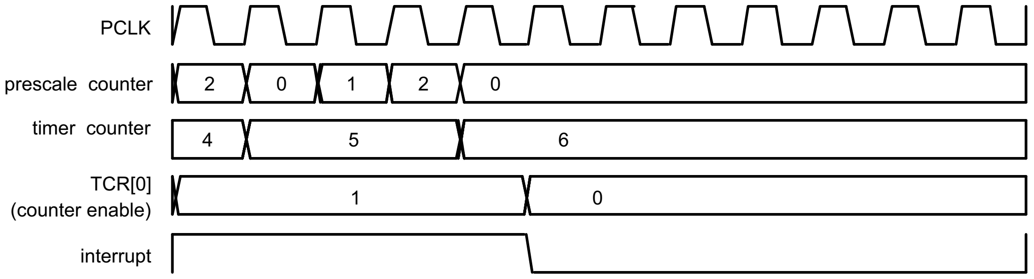


Fig 77. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

20.7.1 Rules for single edge controlled PWM outputs

1. All single edge controlled PWM outputs go LOW at the beginning of each PWM cycle (timer is set to zero) unless their match value is equal to zero.
2. Each PWM output will go HIGH when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM cycle length), the PWM output remains continuously LOW.
3. If a match value larger than the PWM cycle length is written to the match register, and the PWM signal is HIGH already, then the PWM signal will be cleared with the start of the next PWM cycle.
4. If a match register contains the same value as the timer reset value (the PWM cycle length), then the PWM output will be reset to LOW on the next clock tick after the timer reaches the match value. Therefore, the PWM output will always consist of a one clock tick wide positive pulse with a period determined by the PWM cycle length (i.e. the timer reload value).
5. If a match register is set to zero, then the PWM output will go to HIGH the first time the timer goes back to zero and will stay HIGH continuously.

Note: When the match outputs are selected to perform as PWM outputs, the timer reset (MRnR) and timer stop (MRnS) bits in the Match Control Register MCR must be set to zero except for the match register setting the PWM cycle length. For this register, set the MRnR bit to one to enable the timer reset when the timer value matches the value of the corresponding match register.

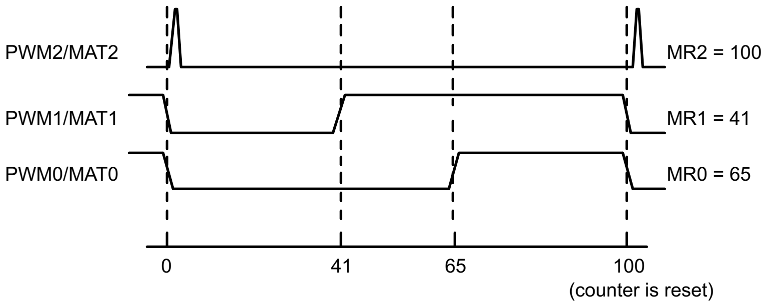


Fig 78. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

UM10732

Chapter 21: LPC11U6x/E6x Real-Time Clock (RTC)

Rev. 1.8 — 19 December 2016 **User manual**

21.1 How to read this chapter

The RTC is available on all parts.

21.2 Features

- The RTC resides in a separate always-on voltage domain with battery back-up. The RTC uses an independent oscillator, also located in the always-on voltage domain.
- 32-bit, 1 Hz RTC counter and associated match register for alarm generation.
- Separate 16-bit high-resolution/wake-up timer clocked at 1.024 kHz for 0.977 ms resolution with a more that one minute maximum time-out period.
- RTC alarm and high-resolution/wake-up timer time-out each generate independent interrupt requests. Either time-out can wake up the part from any of the low power modes, including Deep power-down.

21.3 Basic configuration

Configure the RTC as follows:

- Use the SYSAHBCLKCTRL register (Table 40) to enable the clock to the RTC register interface and peripheral clock.
- RTC software reset supported. See Table 330.
- The RTC provides two interrupts which are ORed in the NVIC interrupt #25:
  a. Interrupt raised on a match of the RTC 1 Hz counter (RTC_ALARM).
  b. Interrupt raised on a match of the RTC 1.024kHz counter (RTC_WAKE).
- To enable the RTC interrupts for waking up from Deep-sleep and Power-down modes, enable the interrupt in the STARTLOGIC1 register (Table 66) and the NVIC.
- To enable the RTC interrupts for waking up from Deep power-down, enable the appropriate RTC clock and wake-up in the RTC CTRL register (Table 330).
- The RTC has no external pins.
- The RTC oscillator is always running when V_{DD} or VBAT are present. The 1 Hz output is enabled in the RTC CTRL register (RTC_EN bit). Once the 1 Hz output is enabled, you can enable the 1.024 KHz output for the high-resolution wake-up timer. The 32.768 kHz output of the RTC oscillator can be enabled in the SYSCON to be used as the main clock.

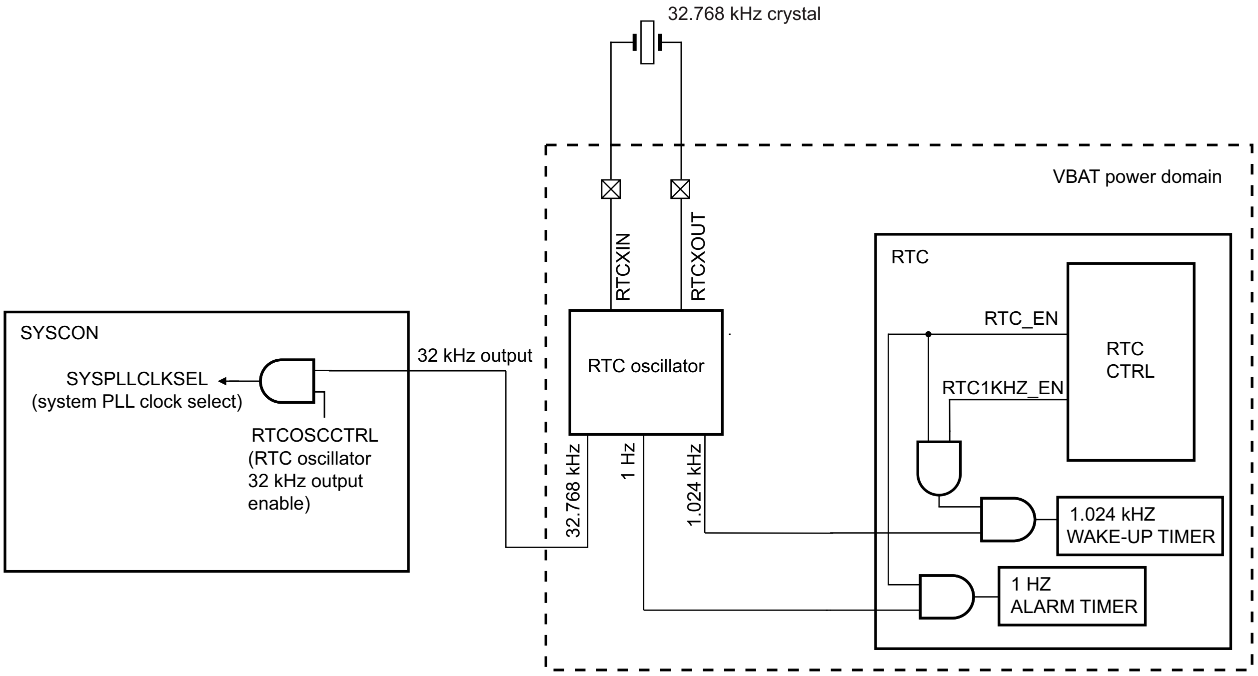


Fig 79. RTC clocking

21.3.1 RTC timers

The RTC contains two timers:

1. The main RTC timer. This 32-bit timer uses a 1 Hz clock and is intended to run continuously as a real-time clock. When the timer value reaches a match value, an interrupt is raised. The alarm interrupt can also wake up the part from any low power mode if enabled.
2. The high-resolution/wake-up timer. This 16-bit timer uses a 1.024 kHz clock and operates as a one-shot down timer. Once the timer is loaded, it starts counting down to 0 at which point an interrupt is raised. The interrupt can wake up the part from any low power mode if enabled. This timer is intended to be used for timed wake-up from Deep-sleep, power-down, or Deep power-down modes. The high-resolution wake-up timer can be disabled to conserve power if not used.

21.4 General description

21.4.1 Real-time clock

The real-time clock is a 32-bit up-counter which can be cleared or initialized by software. Once enabled, it counts continuously at a 1 Hz clock rate as long as the RTC module remains powered and enabled.

The main purpose of the RTC is to count seconds and generate an alarm interrupt to the processor whenever the counter value equals the value programmed into the associated 32-bit match register.

If the part is in one of the reduced-power modes (deep-sleep, power-down, deep power-down) an RTC alarm interrupt can also wake up the part to exit the power mode and begin normal operation.

21.4.2 High-resolution/wake-up timer

The time interval required for many applications, including waking the part up from a low-power mode, will often demand a greater degree of resolution than the one-second minimum interval afforded by the main RTC counter. For these applications, a higher frequency secondary timer has been provided.

This secondary timer is an independent, stand-alone wake-up or general-purpose timer for timing intervals of up to 64 seconds with approximately one millisecond of resolution.

The High-Resolution/Wake-up Timer is a 16-bit down counter which is clocked at a 1.024 kHz rate when it is enabled. Writing any non-zero value to this timer will automatically enable the counter and launch a countdown 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 High-Resolution/Wake-up Timer will turn on, count from the pre-loaded value down to zero, generate an interrupt and/or a wake-up command, and then turn itself off until re-launched by a subsequent software write.

21.4.3 RTC power domain

The RTC module and the 1 Hz/1.024 kHz clock that drives it, reside in the battery backup always-on voltage domain. As a result, the RTC will continue operating in deep power-down mode when power is removed from the rest of the part. The RTC will also continue to operate in the event that power fails, until the backup battery runs out.

21.5 Register description

Reset Values pertain to initial power-up of the always-on power domain or when an RTC software reset is applied (except where noted). This block is not initialized by a standard POR, pad reset, or by any other system reset.

Table 329. Register overview: RTC (base address 0x4002 4000)

Name	Access	Offset	Description	Reset value	Reference
CTRL	R/W	0x000	RTC control register	0xF	Table 330
MATCH	R/W	0x004	RTC match register	0xFFFF	Table 331
COUNT	R/W	0x008	RTC counter register	0	Table 332
WAKE	R/W	0x00C	RTC high-resolution/wake-up timer control register	0	Table 333

21.5.1 RTC CTRL register

This register controls which clock the RTC uses (1.024 kHz or 1 Hz) and enables the two RTC interrupts to wake up the part from Deep power-down. To wake up the part from Deep-sleep or Power-down modes, enable the RTC interrupts in the system control block STARTLOGIC1 register.

Table 330. RTC control register (CTRL, address 0x4002 4000) bit description

Bit	Symbol	Value	Description	Reset value
0	SWRESET		Software reset control	1
		0	Not in reset. The RTC is not held in reset. This bit must be cleared prior to configuring or initiating any operation of the RTC.	
		1	In reset. The RTC is held in reset. All register bits within the RTC will be forced to their reset value except the OFD bit. This bit must be cleared before writing to any register in the RTC - including writes to set any of the other bits within this register. Do not attempt to write to any bits of this register at the same time that the reset bit is being cleared. **Remark:** This bit may also serve as a Power Fail Detect flag for the always-on voltage domain.	
1	OFD		Oscillator fail detect status.	1
		0	Run. The RTC oscillator is running properly. Writing a 0 has no effect.	
		1	Fail. RTC oscillator fail detected. Clear this flag after the following power-up. Writing a 1 clears this bit.	
2	ALARM1HZ		RTC 1 Hz timer alarm flag status.	1
		0	No match. No match has occurred on the 1 Hz RTC timer. Writing a 0 has no effect.	
		1	Match. A match condition has occurred on the 1 Hz RTC timer. This flag generates an RTC alarm interrupt request RTC_ALARM which can also wake up the part from any low power mode. Writing a 1 clears this bit.	
3	WAKE1KHZ		RTC 1.024 kHz timer wake-up flag status.	1
		0	Run. The RTC 1.024 kHz timer is running. Writing a 0 has no effect.	
		1	Time-out. The 1 kHz high-resolution/wake-up timer has timed out. This flag generates an RTC wake-up interrupt request RTC-WAKE which can also wake up the part from any low power mode. Writing a 1 clears this bit.	
4	ALARMDPD_EN		RTC 1 Hz timer alarm enable for Deep power-down.	0
		0	Disable. A match on the 1 Hz RTC timer will not bring the part out of Deep power-down mode.	
		1	Enable. A match on the 1 Hz RTC timer bring the part out of Deep power-down mode.	
5	WAKEDPD_EN		RTC 1.024 kHz timer wake-up enable for Deep power-down.	0
		0	Disable. A match on the 1.024 kHz RTC timer will not bring the part out of Deep power-down mode.	
		1	Enable. A match on the 1.024 kHz RTC timer bring the part out of Deep power-down mode.	

Table 330. RTC control register (CTRL, address 0x4002 4000) bit description

Bit	Symbol	Value	Description	Reset value
6	RTC1KHZ_EN		RTC 1.024 kHz clock enable. This bit can be set to 0 to conserve power if the 1.024 kHz timer is not used. This bit has no effect when the RTC is disabled (bit 7 of this register is 0).	0
		0	Disable. A match on the 1.024 kHz RTC timer will not bring the part out of Deep power-down mode.	
		1	Enable. The 1.024 kHz RTC timer is enabled.	
7	RTC_EN		RTC enable.	0
		0	Disable. The RTC 1 Hz and 1.024 kHz clocks are shut down and the RTC operation is disabled. This bit should be 0 when writing to load a value in the RTC counter register.	
		1	Enable. The 1 Hz RTC clock is running and RTC operation is enabled. You must set this bit to initiate operation of the RTC. The first clock to the RTC counter occurs 1 s after this bit is set. To also enable the high-resolution, 1.024 kHz clock, set bit 6 in this register.	
31:8	-		Reserved	0

21.5.2 RTC match register

Table 331. RTC match register (MATCH, address 0x4002 4004) bit description

Bit	Symbol	Description	Reset value
31:0	MATVAL	Contains the match value against which the 1 Hz RTC timer will be compared to generate set the alarm flag RTC_ALARM and generate an alarm interrupt/wake-up if enabled.	0xFFFF

21.5.3 RTC counter register

Table 332. RTC counter register (COUNT, address 0x4002 4008) bit description

Bit	Symbol	Description	Reset value
31:0	VAL	A read reflects the current value of the main, 1 Hz RTC timer. A write loads a new initial value into the timer. The RTC counter will count up continuously at a 1 Hz rate once the RTC Software Reset is removed (by clearing bit 0 of the CTRL register). **Remark:** Only write to this register when the RTC1HZ_EN bit in the RTC CTRL Register is 0. The counter increments one second after the RTC1HZ_EN bit is set.	0

21.5.4 RTC high-resolution/wake-up register

Table 333. RTC high-resolution/wake-up register (WAKE, address 0x4002 400C) bit description

Bit	Symbol	Description	Reset value
15:0	VAL	A read reflects the current value of the high-resolution/wake-up timer. A write pre-loads a start count value into the wake-up timer and initializes a count-down sequence. Do not write to this register while counting is in progress.	0
31:16	-	Reserved.	

UM10732

Chapter 22: LPC11U6x/E6x Windowed Watchdog Timer (WWDT)

Rev. 1.8 — 19 December 2016 **User manual**

22.1 How to read this chapter

The WWDT is available on all parts.

22.2 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 can be selected as the Internal High frequency oscillator (IRC) or the WatchDog oscillator.
- The Watchdog timer can be configured to run in Deep-sleep or Power-down mode when using the watchdog oscillator as the clock source.
- Debug mode.

22.3 Basic configuration

The WWDT is configured through the following registers:

- Power to the register interface (WWDT PCLK clock): In the SYSAHBCLKCTRL register, set bit 15 in Table 40.
- Enable the WWDT clock source (the watchdog oscillator or the IRC) in the PDRUNCFG register (Table 69).
- For waking up from a WWDT interrupt, enable the watchdog interrupt for wake-up in the STARTERP1 register (Table 66).

22.4 General description

The Watchdog consists of a fixed (divide by 4) pre-scaler and a 24 bit counter which decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum Watchdog interval is ($T_{WDCLK} \times 256 \times 4$) and the maximum Watchdog interval is ($T_{WDCLK} \times 2^{24} \times 4$) in multiples of ($T_{WDCLK} \times 4$). The Watchdog should be used in the following manner:

- Set the Watchdog timer constant reload value in 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.

22.4.1 Block diagram

The block diagram of the Watchdog is shown below in the Figure 80. The synchronization logic (PCLK - WDCLK) is not shown in the block diagram.

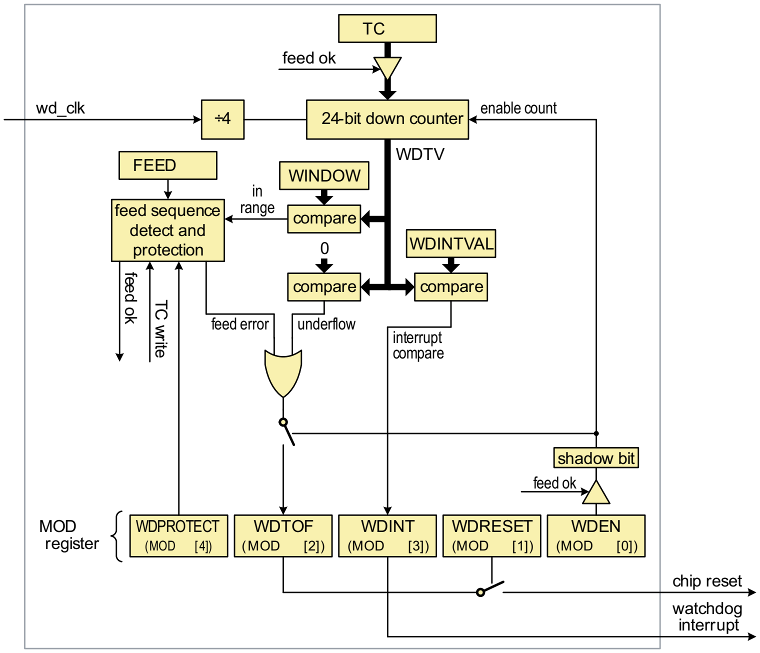


Fig 80. Watchdog block diagram

22.4.2 Applications

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 and/or will be 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.

22.4.3 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 4). The WDCLK is used for the watchdog timer counting and is derived from the wdt_clk in Figure 4. Either the IRC or the watchdog oscillator can be used as wdt_clk in Active mode, Sleep mode, and Deep-sleep modes. In Power-down mode only the watchdog oscillator is available.

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.

22.4.4 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:

- Accidental overwrite of the WWDT clock source
- Changing the WWDT clock source
- Changing the WWDT reload value

22.4.4.1 Accidental overwrite of the WWDT clock

If bit 31 of the WWDT CLKSEL register (Table 340) is set, writes to bit 0 of the CLKSEL register, the clock source select bit, will be ignored and the clock source will not change.

22.4.4.2 Changing the WWDT clock source

If bit 5 in the WWDT MOD register is set, the current clock source as selected in the CLKSEL register is locked and can not be changed either by software or by hardware when Sleep, Deep-sleep or Power-down modes are entered. Therefore, the user must ensure that the appropriate WWDT clock source for each power mode is selected **before** setting bit 5 in the MOD register:

- Active or Sleep modes: Both the IRC or the watchdog oscillator are allowed.
- Deep-sleep mode: Both the IRC and the watchdog oscillator are allowed. However, using the IRC during Deep-sleep mode will increase the power consumption. To minimize power consumption, use the watchdog oscillator as clock source.
- Power-down mode: Only the watchdog oscillator is allowed as clock source for the WWDT. Therefore, before setting bit 5 and locking the clock source, the WWDT clock source must be set to the watchdog oscillator. Otherwise, the part may not be able to enter Power-down mode.
- Deep power-down mode: No clock locking mechanisms are in effect as neither the WWDT nor any of the 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 77).

The clock source lock mechanism can only be disabled by a reset of any type.

22.4.4.3 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.

22.5 Register description

Table 334. Register overview: Watchdog timer (base address 0x4000 4000)

Name	Access	Address offset	Description	Reset Value[1]	Reference
MOD	R/W	0x000	Watchdog mode register. This register contains the basic mode and status of the Watchdog Timer.	0	Table 335
TC	R/W	0x004	Watchdog timer constant register. This 24-bit register determines the time-out value.	0xFF	Table 337
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 338
TV	RO	0x00C	Watchdog timer value register. This 24-bit register reads out the current value of the Watchdog timer.	0xFF	Table 339
CLKSEL	R/W	0x010	Watchdog clock select register.	0	Table 340
WARNINT	R/W	0x014	Watchdog Warning Interrupt compare value.	0	Table 341
WINDOW	R/W	0x018	Watchdog Window compare value.	0xFF FFFF	Table 342

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

22.5.1 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 335. 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 rewritten with a 0.	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 rewritten with a 0.	0
		0	A watchdog timeout will not cause a chip reset.	
		1	A watchdog timeout will cause a chip reset.	
2	WDTOF		Watchdog time-out flag. Set when the watchdog timer times out, by a feed error, or by events associated with WDPROTECT. Cleared by software. Causes a chip reset if WDRESET = 1.	0 (only after external reset)
3	WDINT		Warning interrupt flag. Set when the timer reaches the value in WDWARNINT. Cleared by software.	0

Table 335. Watchdog mode register (MOD, 0x4000 4000) bit description

Bit	Symbol	Value	Description	Reset value
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 clock source selected by bit 0 of the WDCLKSRC register and also prevents switching to a clock source that is disabled or powered down. This bit can be set once by software and is only cleared by any reset. **Remark:** If this bit is one and the WWDT clock source is the IRC when Deep-sleep or Power-down modes are entered, the IRC remains running thereby increasing power consumption in Deep-sleep mode and potentially preventing the part from entering Power-down mode correctly.	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 1 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 or the IRC can be selected to keep running in Sleep and Deep-sleep modes. In Power-down mode, only the watchdog oscillator is allowed. 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.

Table 336. 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.

22.5.2 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 337. 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

22.5.3 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 practise to disable interrupts around a feed sequence, if the application is such that some/any 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 338. 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

22.5.4 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 339. 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

22.5.5 Watchdog Clock Select register

The LOCK bit in this register prevents software from changing the clock source inadvertently. Once the LOCK bit is set, software cannot change the clock source until this register has been reset from any reset source.

Table 340. Watchdog Clock Select register (CLKSEL, 0x4000 4010) bit description

Bit	Symbol	Value	Description	Reset Value
0	CLKSEL		Selects source of WDT clock	0
		0	IRC	
		1	Watchdog oscillator (WDOSC)	
30:1	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
31	LOCK		If this bit is set to one, writing to this register does not affect bit 0 (that is the clock source cannot be changed). The clock source can only by changed after a reset from any source.	0

22.5.6 Watchdog Timer Warning Interrupt register

The WDWARNINT register determines the watchdog timer counter value that will generate a watchdog interrupt. When the watchdog timer counter matches the value defined by WDWARNINT, an interrupt will be generated after the subsequent WDCLK.

A match of the watchdog timer counter to WDWARNINT occurs when the bottom 10 bits of the counter have the same value as the 10 bits of WARNINT, and the remaining upper bits of the counter are all 0. This gives a maximum time of 1,023 watchdog timer counts (4,096 watchdog clocks) for the interrupt to occur prior to a watchdog event. If WARNINT is 0, the interrupt will occur at the same time as the watchdog event.

Table 341. 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

22.5.7 Watchdog Timer Window register

The WDWINDOW 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 WDWINDOW, a watchdog event will occur.

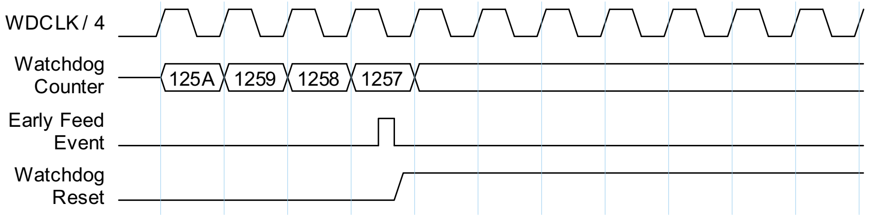
WDWINDOW resets to the maximum possible WDTV value, so windowing is not in effect.

Table 342. 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

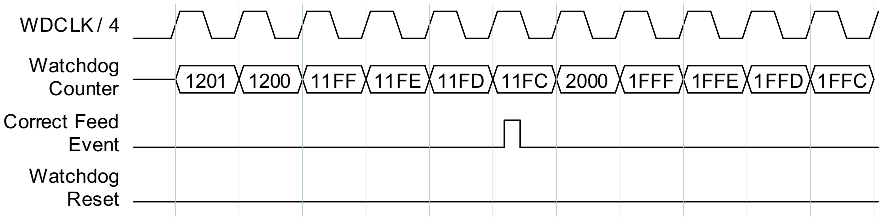
22.6 Watchdog timing examples

The following figures illustrate several aspects of Watchdog Timer operation.



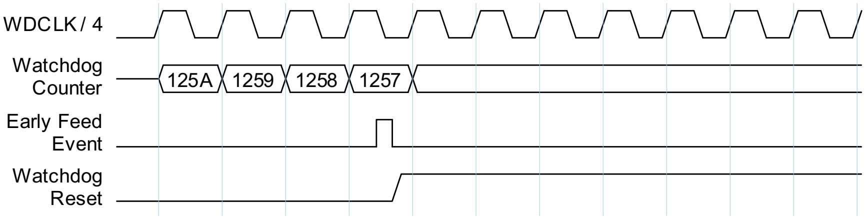
Conditions:
WINDOW = 0x1200
WARNINT = 0x3FF
TC = 0x2000

Fig 81. Early Watchdog Feed with Windowed Mode Enabled



Conditions:
WDWINDOW = 0x1200
WDWARNINT = 0x3FF
WDTC = 0x2000

Fig 82. Correct Watchdog Feed with Windowed Mode Enabled



Conditions:
WINDOW = 0x1200
WARNINT = 0x3FF
TC = 0x2000

Fig 83. Watchdog Warning Interrupt

UM10732

Chapter 23: LPC11U6x/E6x Cyclic Redundancy Check (CRC) engine

Rev. 1.8 — 19 December 2016 **User manual**

23.1 How to read this chapter

The CRC engine is available on all parts.

23.2 Features

- Supports three common polynomials CRC-CCITT, CRC-16, and CRC-32.
  - CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$
  - CRC-16: $x^{16} + x^{15} + x^2 + 1$
  - CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Bit order reverse and 1’s complement programmable setting for input data and CRC sum.
- Programmable seed number setting.
- Accept any size of data width per write: 8, 16 or 32-bit.
  - 8-bit write: 1-cycle operation
  - 16-bit write: 2-cycle operation (8-bit x 2-cycle)
  - 32-bit write: 4-cycle operation (8-bit x 4-cycle)

23.3 Basic configuration

Enable the clock to the CRC engine in the SYSAHBCLKCTRL register (Table 40).

23.4 Pin description

The CRC engine has no configurable pins.

23.5 General description

The Cyclic Redundancy Check (CRC) generator with programmable polynomial settings supports several CRC standards commonly used.

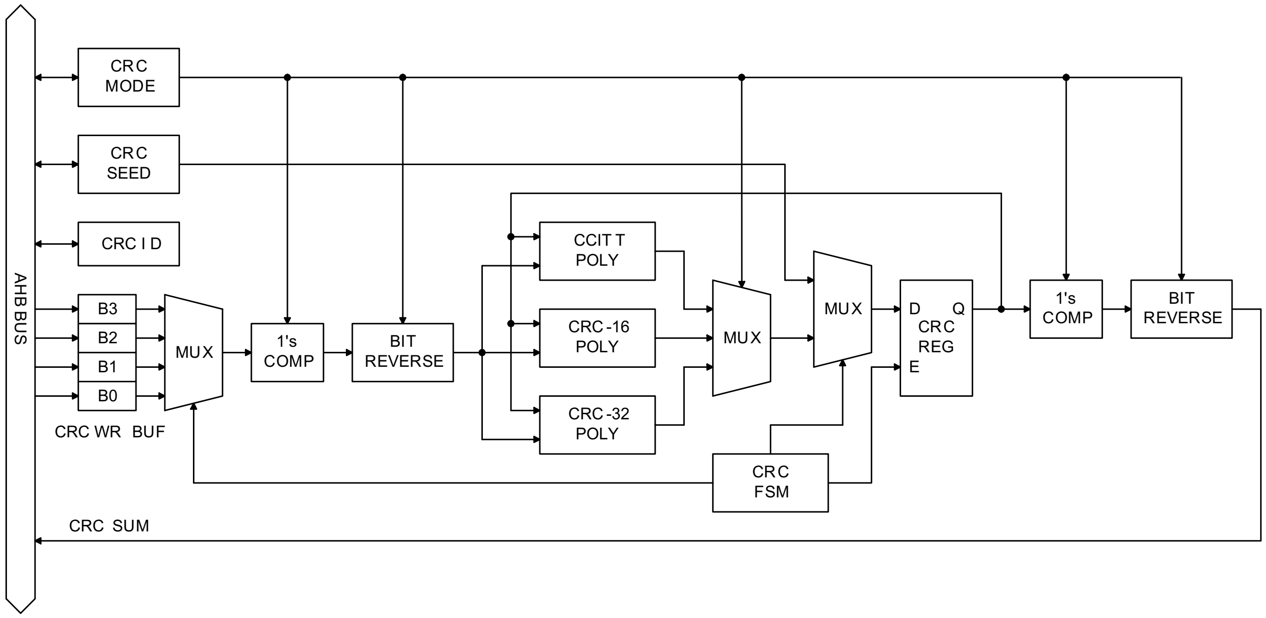


Fig 84. CRC block diagram

23.6 Register description

Table 343. 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 344
SEED	R/W	0x004	CRC seed register	0x0000 FFFF	Table 345
SUM	RO	0x008	CRC checksum register	0x0000 FFFF	Table 346
WR_DATA	WO	0x008	CRC data register	-	Table 347

23.6.1 CRC mode register

Table 344. CRC mode register (MODE, address 0x5000 0000) bit description

Bit	Symbol	Description	Reset value
1:0	CRC_POLY	CRC polynom: 1X= CRC-32 polynomial 01= CRC-16 polynomial 00= CRC-CCITT polynomial	00
2	BIT_RVS_WR	Data bit order: 1= Bit order reverse for CRC_WR_DATA (per byte) 0= No bit order reverse for CRC_WR_DATA (per byte)	0
3	CMPL_WR	Data complement: 1= 1’s complement for CRC_WR_DATA 0= No 1’s complement for CRC_WR_DATA	0
4	BIT_RVS_SUM	CRC sum bit order: 1= Bit order reverse for CRC_SUM 0= No bit order reverse for CRC_SUM	0
5	CMPL_SUM	CRC sum complement: 1= 1’s complement for CRC_SUM 0=No 1’s complement for CRC_SUM	0
31:6	Reserved	Always 0 when read	0x0000000

23.6.2 CRC seed register

Table 345. CRC seed register (SEED, address 0x5000 0004) bit description

Bit	Symbol	Description	Reset value
31:0	CRC_SEED	A write access to this register will load CRC seed value to CRC_SUM register with selected bit order and 1’s complement pre-processes. **Remark:** A write access to this register will overrule the CRC calculation in progresses.	0x0000 FFFF

23.6.3 CRC checksum register

This register is a Read-only register containing the most recent checksum. The read request to this register is automatically delayed by a finite number of wait states until the results are valid and the checksum computation is complete.

Table 346. CRC checksum register (SUM, address 0x5000 0008) bit description

Bit	Symbol	Description	Reset value
31:0	CRC_SUM	The most recent CRC sum can be read through this register with selected bit order and 1’s complement post-processes.	0x0000 FFFF

23.6.4 CRC data register

This register is a Write-only register containing the data block for which the CRC sum will be calculated.

Table 347. CRC data register (WR_DATA, address 0x5000 0008) bit description

Bit	Symbol	Description	Reset value
31:0	CRC_WR_DATA	Data written to this register will be taken to perform CRC calculation with selected bit order and 1’s complement pre-process. Any write size 8, 16 or 32-bit are allowed and accept back-to-back transactions.	-

23.7 Functional description

The following sections describe the register settings for each supported CRC standard:

23.7.1 CRC-CCITT set-up

Polynomial = $x^{16} + x^{12} + x^5 + 1$
Seed Value = 0xFFFF
Bit order reverse for data input: NO
1's complement for data input: NO
Bit order reverse for CRC sum: NO
1's complement for CRC sum: NO
CRC_MODE = 0x0000 0000
CRC_SEED = 0x0000 FFFF

23.7.2 CRC-16 set-up

Polynomial = $x^{16} + x^{15} + x^2 + 1$
Seed Value = 0x0000
Bit order reverse for data input: YES
1's complement for data input: NO
Bit order reverse for CRC sum: YES
1's complement for CRC sum: NO
CRC_MODE = 0x0000 0015
CRC_SEED = 0x0000 0000

23.7.3 CRC-32 set-up

Polynomial = $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$

Seed Value = 0xFFFF FFFF

Bit order reverse for data input: YES

1's complement for data input: NO

Bit order reverse for CRC sum: YES

1's complement for CRC sum: YES

CRC_MODE = 0x0000 0036

CRC_SEED = 0xFFFF FFFF

UM10732

Chapter 24: LPC11U6x/E6x System tick timer (SysTick)

Rev. 1.8 — 19 December 2016 **User manual**

24.1 How to read this chapter

The system tick timer (SysTick timer) is part of the ARM Cortex-M0+ core and is available on all parts.

24.2 Basic configuration

The system tick timer is configured using the following registers:

1. Pins: The system tick timer uses no external pins.
2. Power: The system tick timer is enabled through the SysTick control register (Table 452). The system tick timer clock is fixed to half the frequency of the system clock.
3. Enable the clock source for the SysTick timer in the SYST_CSR register (Table 452).

24.3 Features

- Simple 24-bit timer.
- Uses dedicated exception vector.
- Clocked internally by the system clock or the system clock/2.

24.4 General description

The block diagram of the SysTick timer is shown below in the Figure 85.

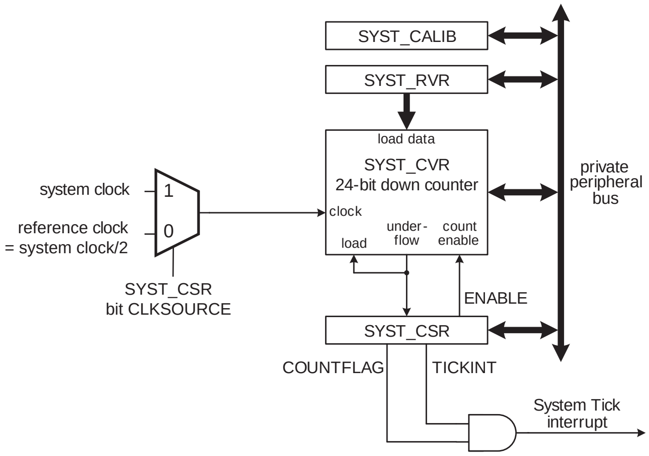


Fig 85. System tick timer block diagram

The SysTick timer is an integral part of the Cortex-M0+. The SysTick timer is intended to generate a fixed 10 millisecond interrupt for use by an operating system or other system management software.

Since the SysTick timer is a part of the Cortex-M0, it facilitates porting of software by providing a standard timer that is available on Cortex-M0+ based devices. The SysTick timer can be used for:

- An RTOS tick timer which fires at a programmable rate (for example 100 Hz) and invokes a SysTick routine.
- A high-speed alarm timer using the core clock.
- A simple counter. Software can use this to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNTFLAG bit-field in the control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

Refer to the *Cortex-M0+ User Guide* for details.

24.5 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.

Table 348. Register overview: SysTick timer (base address 0xE000 E000)

Name	Access	Address offset	Description	Reset value[1]	Reference
SYST_CSR	R/W	0x010	System Timer Control and status register	0x000 0000	Table 349
SYST_RVR	R/W	0x014	System Timer Reload value register	0	Table 350
SYST_CVR	R/W	0x018	System Timer Current value register	0	Table 351
SYST_CALIB	R/W	0x01C	System Timer Calibration value register	0x4	Table 352

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

24.5.1 System Timer Control and status register

The SYST_CSR register contains control information for the SysTick timer and provides a status flag. This register is part of the ARM Cortex-M0+ core system timer register block.

This register determines the clock source for the system tick timer.

Table 349. 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

Table 349. SysTick Timer Control and status register (SYST_CSR, 0xE000 E010) bit description

Bit	Symbol	Description	Reset value
15:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
16	COUNTFLAG	Returns 1 if the SysTick timer counted to 0 since the last read of this register.	0
31:17	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

24.5.2 System Timer Reload value register

The SYST_RVR register is set to the value that will be loaded into the SysTick timer whenever it counts down to zero. This register is loaded by software as part of timer initialization. The SYST_CALIB register may be read and used as the value for SYST_RVR register if the CPU is running at the frequency intended for use with the SYST_CALIB value.

Table 350. 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

24.5.3 System Timer Current value register

The SYST_CVR register returns the current count from the System Tick counter when it is read by software.

Table 351. 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 SYST_CSR.	0
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

24.5.4 System Timer Calibration value register

The value of the SYST_CALIB register is driven by the value of the SYSTCKCAL register in the system configuration block (see Table 59).

Table 352. System Timer Calibration value register (SYST_CALIB, 0xE000 E01C) bit description

Bit	Symbol	Value	Description	Reset value
23:0	TENMS		Calibration value.	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		Calibration value.	0
31	NOREF		Calibration value.	0

24.6 Functional description

The SysTick timer is a 24-bit timer that counts down to zero and generates an interrupt. The intent is to provide a fixed 10 millisecond time interval between interrupts. The SysTick timer is clocked from the CPU clock (the system clock, see Figure 4) 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 50 MHz.

24.7 Example timer calculations

To use the system tick timer, do the following:

1. Program the SYST_RVR register with the reload value RELOAD to obtain the desired time interval.
2. Clear the SYST_CVR register by writing to it. This ensures that the timer will count from the SYST_RVR value rather than an arbitrary value when the timer is enabled.
3. Program the SYST_SCR register with the value 0x7 which enables the SysTick timer and the SysTick timer interrupt.

The following example illustrates selecting the SysTick timer reload value to obtain a 10 ms time interval with the LPC11U3x/2x/1x system clock set to 50 MHz.

Example (system clock = 50 MHz)

The system tick clock = system clock = 50 MHz. Bit CLKSOURCE in the SYST_CSR register set to 1 (system clock).

RELOAD = (system tick clock frequency × 10 ms) −1 = (50 MHz × 10 ms) −1 = 500000 −1 = 499999 = 0x0007A11F.

UM10732

Chapter 25: LPC11U6x/E6x Flash controller

Rev. 1.8 — 19 December 2016 **User manual**

25.1 How to read this chapter

The flash controller is identical on all parts.

25.2 Features

- Controls flash access time.
- Provides registers for flash signature generation.

25.3 General description

The flash controller is accessible for programming flash wait states and for generating the flash signature.

25.4 Register description

Table 353. Register overview: FMC (base address 0x4003 C000)

Name	Access	Address offset	Description	Reset value	Reference
FLASHCFG	R/W	0x010	Flash configuration register	-	Table 354
FMSSTART	R/W	0x020	Signature start address register	0	Table 355
FMSSTOP	R/W	0x024	Signature stop-address register	0	Table 356
FMSW0	R	0x02C	Signature word	-	Table 357

25.4.1 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 354. Flash configuration register (FLASHCFG, address 0x4003 C010) bit description

Bit	Symbol	Value	Description	Reset value
1:0	FLASHTIM		Flash memory access time. FLASHTIM +1 is equal to the number of system clocks used for flash access.	0x2
		0x0	1 system clock flash access time (for system clock frequencies of up to 20 MHz).	
		0x1	2 system clocks flash access time (for system clock frequencies of up to 40 MHz).	
		0x2	3 system clocks flash access time (for system clock frequencies of up to 50 MHz).	
		0x3	Reserved.	
31:2	-	-	Reserved. **User software must not change the value of these bits. Bits 31:2 must be written back exactly as read**.	-

25.4.2 Flash signature start address register

Table 355. Flash Module Signature Start register (FMSSTART, 0x4003 C020) bit description

Bit	Symbol	Description	Reset value
16:0	START	Signature generation start address (corresponds to AHB byte address bits[20:4]).	0
31:17	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

25.4.3 Flash signature stop address register

Table 356. Flash Module Signature Stop register (FMSSTOP, 0x4003 C024) bit description

Bit	Symbol	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
17	STRTBIST	When this bit is written to 1, signature generation starts. At the end of signature generation, this bit is automatically cleared.	0
31:18	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

25.4.4 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 25.5.1.

Table 357. FMSW0 register bit description (FMSW0, address: 0x4003 C02C)

Bit	Symbol	Description	Reset value
31:0	SIG	32-bit signature.	-

25.5 Functional description

25.5.1 Flash signature generation

The flash module contains a built-in signature generator. This generator can produce a 128-bit signature from a range of flash memory. A typical usage is to verify the flashed contents against a calculated signature (example, during programming).

The address range for generating a signature must be aligned on flash-word boundaries, that is, 128-bit boundaries. Once started, signature generation completes independently. While signature generation is in progress, the flash memory cannot be accessed for other purposes, and an attempted read will cause a wait state to be asserted until signature generation is complete. Code outside of the flash (example, 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.

25.5.1.1 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 355 and Table 356 show the bit assignments in the FMSSTART and FMSSTOP registers respectively.

25.5.1.2 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.

25.5.1.3 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
```

UM10732

Chapter 26: LPC11U6x/E6x Boot ROM

Rev. 1.8 — 19 December 2016 **User manual**

26.1 How to read this chapter

The boot ROM is available on all parts. USB boot and USB ROM API functions are only available on LPC11U6x.

26.2 Features

- 32 KB on-chip boot ROM
- Contains the boot loader with In-System Programming (ISP) facility and the following APIs:
  - Boot loader.
  - Flash In-Application Programming (IAP) and In-System Programming (ISP).
  - Power profiles for optimizing power consumption and system performance
  - USART drivers
  - I2C drivers
  - USB drivers.
  - Power profiles.

26.3 Basic configuration

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

26.4 Pin description

The ISP command handler uses the USART0 interface to communicate via the serial port.

Table 358. Pins in ISP mode

Function	Pin	Description
ISP entry pin	PIO0_1	A LOW level on this pin during reset starts the ISP command handler or the USB device enumeration.
USB device enumeration select	PIO0_3	A LOW level on this pin during reset starts the ISP command handler. A HIGH level during reset starts the USB device enumeration.
USART0 receive	PIO0_18	UART receive in ISP mode via the USART0 peripheral
USART0 transmit	PIO0_19	UART transmit in ISP mode via the USART0 peripheral

26.5 General description

26.5.1 Boot loader

The bootloader code is executed every time the part is powered on or reset (see Figure 86). The loader can execute the ISP command handler or the user application code. The chip interprets a LOW level during reset at the ISP entry pin as an external hardware request to start the ISP command handler (or the USB device handler - see Section 27.7.3) without checking for a valid user code first.

Assuming that power supply pins are at their nominal levels when the rising edge on $\overline{\text{RESET}}$ pin is generated, it may take up to 3 ms before the ISP entry pin is sampled and the decision whether to continue with user code or ISP handler is made. The boot loader performs the following steps (see Figure 86):

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

The state of PIO0_3 determines whether the UART or USB interface will be used (see Section 27.7.3):

- If PIO0_3 is sampled HIGH, the bootloader connects the part as a MSC USB device to a PC host. The part's flash memory space is represented as a drive in the host's operating system.
- If PIO0_3 is sampled LOW, the bootloader configures the UART serial port using pins PIO0_18 and PIO0_19 for RXD and TXD and calls the ISP command handler.

Remark: The sampling of the ISP entry pin can be disabled through programming flash location 0x0000 02FC (see Section 27.4.2.1).

26.5.2 Memory map after any reset

The boot block is 32 KB in size and is located in the memory region starting from the address 0x1FFF 0000. The bootloader is designed to run from this memory area, but both the ISP and IAP software use parts of the on-chip RAM. The RAM usage is described later in this chapter. The interrupt vectors residing in the boot block of the on-chip flash memory also become active after reset, i.e., the bottom 512 bytes of the boot block are also visible in the memory region starting from the address 0x0000 0000.

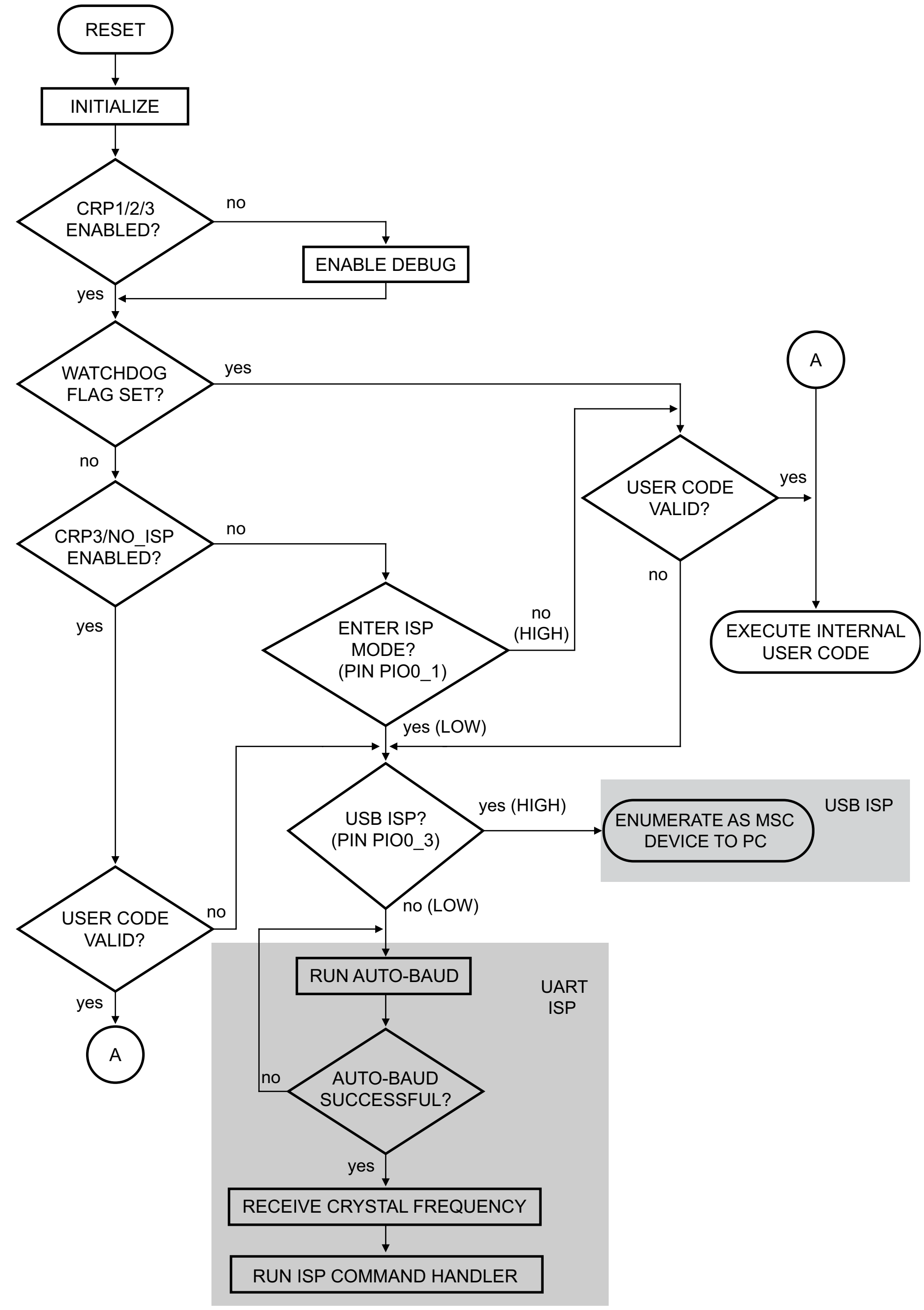
26.5.3 Boot process

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

The reserved ARM 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 boot code checks pin PIO0_3 and enumerates as USB MSC device (pin PIO0_3 is HIGH) or enters ISP UART mode (PIO0_3 is LOW).

26.5.4 Boot process flowchart



USB ISP mode is available on LPC11U6x parts only.

Fig 86. Boot process flowchart

26.5.5 ROM-based APIs

Once the part has booted, the user can access several APIs located in the boot ROM to access flash and EEPROM memory, to optimize power consumption, and to run peripherals. The structure of the boot ROM APIs is shown in

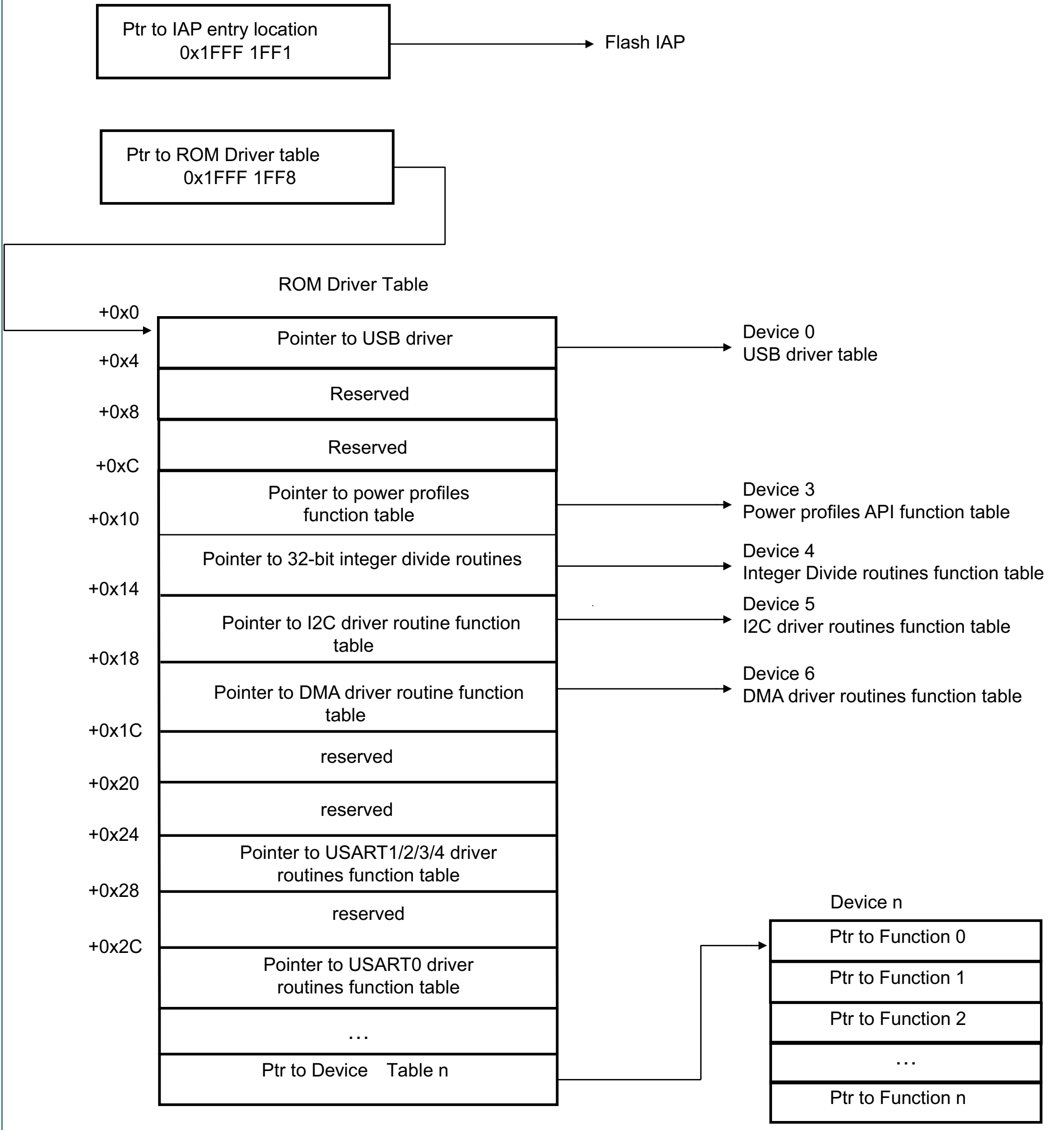


Fig 87. Boot ROM structure

The boot rom structure should be included as follows:

```
typedef struct {
        const uint32_t usbdApiBase;  /*!< USBD API function table base address */
        const uint32_t reserved0;       /*!< Reserved */
        const uint32_t reserved1;       /*!< Reserved */
        const PWRD_API_T *pPWRD; /*!< Power API function table base address */
        const ROM_DIV_API_T *divApiBase; /*!< Divider API function table base address */
        const I2CD_API_T *pI2CD;    /*!< I2C driver API function table base address */
        const DMAD_API_T *pDMAD; /*!< DMA driver API function table base address */
        const uint32_t reserved2;       /*!< Reserved */
        const uint32_t reserved3;       /*!< Reserved */
        const UARTD_API_T *pUARTND; /*!< USART 1/2/3/4 driver API function table base address */
        const uint32_t reserved4;       /*!< Reserved */
        const UARTD_API_T *pUART0D; /*!< USART 0 driver API function table base address */
} LPC_ROM_API_T;

#define ROM_DRIVER_BASE_LOC (0x1FFF1FF8UL)
#define LPC_ROM_API (*(LPC_ROM_API_T * *) LPC_ROM_API_BASE_LOC)
```

Table 359. ROM APIs

API	Description	Reference
Flash IAP	Flash In-Application programming	Table 382
USB driver	USB CDC, HID, mass storage classes	Section 34.3.1
Power profiles API	Configure system clock and power consumption	Table 397
Divide routines	32-bit integer divide routines	Table 402
I2C driver	I2C ROM driver	Table 407
DMA driver	DMA ROM driver	Table 447
UART driver	UART ROM driver for USART1/2/3/4	Table 437
USART0 driver	USART ROM driver for USART0	Table 427

UM10732

Chapter 27: LPC11U6x/E6x Flash/EEPROM ISP/IAP programming

Rev. 1.8 — 19 December 2016 **User manual**

27.1 How to read this chapter

ISP and IAP programming are available for all parts.

27.2 Features

- In-System Programming: In-System programming (ISP) supports programming or reprogramming the on-chip flash memory, using the bootloader software and UART serial port.
- In-Application Programming: In-Application (IAP) programming supports 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.
- The part supports ISP from the USB port through enumeration as a Mass Storage Class (MSC) Device when connected to a USB host interface.

27.3 Pin description

Table 360. Pins in ISP mode

Function	Pin	Description
ISP entry pin	PIO0_1	A LOW level on this pin during reset starts the ISP command handler or the USB device enumeration.
USB device enumeration select	PIO0_3	A LOW level on this pin during reset starts the ISP command handler. A HIGH level during reset starts the USB device enumeration.
USART0 receive	PIO0_18	UART receive in ISP mode via the USART0 peripheral
USART0 transmit	PIO0_19	UART transmit in ISP mode via the USART0 peripheral

27.4 General description

27.4.1 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 part contains up to 256 KB on-chip flash program memory.

The flash memory is divided into 24 x 4 KB and 5 x 32 KB sectors. Individual pages of 256 byte each can be erased using the IAP erase page command.

Table 361. Flash configuration

Sector number	Sector size [KB]	Page number	Address range	LPC11U66/E66 64 KB	LPC11U67/E67 128 KB	LPC11U68/E68 256 KB
0	4	0 -15	0x0000 0000 - 0x0000 0FFF	yes	yes	yes
1	4	16 - 31	0x0000 1000 - 0x0000 1FFF	yes	yes	yes
2	4	32 - 47	0x0000 2000 - 0x0000 2FFF	yes	yes	yes
3	4	48 - 63	0x0000 3000 - 0x0000 3FFF	yes	yes	yes
4	4	64 - 79	0x0000 4000 - 0x0000 4FFF	yes	yes	yes
5	4	80 - 95	0x0000 5000 - 0x0000 5FFF	yes	yes	yes
6	4	96 - 111	0x0000 6000 - 0x0000 6FFF	yes	yes	yes
7	4	112 - 127	0x0000 7000 - 0x00007FFF	yes	yes	yes
8	4	128 - 143	0x0000 8000 - 0x00008FFF	yes	yes	yes
9	4	144 - 159	0x0000 9000 - 0x0000 9FFF	yes	yes	yes
10	4	160 - 175	0x0000 A000 - 0x0000 AFFF	yes	yes	yes
11	4	176 - 191	0x0000 B000 - 0x0000 BFFF	yes	yes	yes
12	4	192 - 207	0x0000 C000 - 0x0000 CFFF	yes	yes	yes
13	4	208 - 223	0x0000 D000 - 0x0000 DFFF	yes	yes	yes
14	4	224 - 239	0x0000 E000 - 0x0000 EFFF	yes	yes	yes
15	4	240 - 255	0x0000 F000 - 0x0000 FFFF	yes	yes	yes
16	4	256 - 271	0x0001 0000 - 0x0001 0FFF	no	yes	yes
17	4	272 - 287	0x0001 1000 - 0x0001 1FFF	no	yes	yes
18	4	288 - 303	0x0001 2000 - 0x0001 2FFF	no	yes	yes
19	4	304 - 319	0x0001 3000 - 0x0001 3FFF	no	yes	yes
20	4	320 - 335	0x0001 4000 - 0x0001 4FFF	no	yes	yes
21	4	336 - 351	0x0001 5000 - 0x0001 5FFF	no	yes	yes
22	4	352 - 367	0x0001 6000 - 0x0001 6FFF	no	yes	yes
23	4	368 - 383	0x0001 7000 - 0x0001 7FFF	no	yes	yes
24	32	384 - 511	0x0001 8000 - 0x0001 FFFF	no	yes	yes
25	32	512 - 639	0x0002 0000 - 0x0002 7FFF	no	no	yes
26	32	640 - 767	0x0002 8000 - 0x0002 FFFF	no	no	yes
27	32	768 - 895	0x0003 0000 - 0x0003 7FFF	no	no	yes
28	32	896 - 1023	0x0003 8000 - 0x0003 FFFF	no	no	yes

27.4.2 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 362. Code Read Protection options

Name	Pattern programmed in 0x0000 02FC	Description
NO_ISP	0x4E69 7370	Prevents sampling of the ISP entry pin for entering ISP mode. The ISP entry pin is available for other uses.
CRP1	0x12345678	Access to chip via the SWD pins is disabled. This mode allows partial flash update using the following ISP commands and restrictions: • Write to RAM command should not access RAM below 0x1000 0300. Access to addresses below 0x1000 0200 is disabled. • Copy RAM to flash command can not write to Sector 0. • Erase command can erase Sector 0 only when all sectors are selected for erase. • Compare command is disabled. • Read Memory command is disabled. This mode is useful when CRP is required and flash field updates are needed but all sectors can not be erased. Since compare command is disabled in case of partial updates the secondary loader should implement checksum mechanism to verify the integrity of the flash.
CRP2	0x87654321	Access to chip via the SWD pins is disabled. The following ISP commands are disabled: • Read Memory • Write to RAM • Go • Copy RAM to flash • Compare When CRP2 is enabled the ISP erase command only allows erasure of all user sectors.
CRP3	0x43218765	Access to chip via the SWD pins is disabled. ISP entry by pulling the ISP entry pin LOW is disabled if a valid user code is present in flash sector 0. This mode effectively disables ISP override using the ISP entry pin. It is up to the user’s application to provide a flash update mechanism using IAP calls or call 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 363. Code Read Protection hardware/software interaction

CRP option	User Code Valid	ISP entry pin at reset	SWD enabled	Part enters ISP mode (UART or USB)	partial flash update in ISP mode
None	No	x	Yes	Yes	Yes
None	Yes	High	Yes	No	NA
None	Yes	Low	Yes	Yes	Yes
CRP1	Yes	High	No	No	NA
CRP1	Yes	Low	No	Yes	Yes
CRP2	Yes	High	No	No	NA
CRP2	Yes	Low	No	Yes	No
CRP3	Yes	x	No	No	NA
CRP1	No	x	No	Yes	Yes
CRP2	No	x	No	Yes	No
CRP3	No	x	No	Yes	No

Table 364. 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.

27.4.2.1 ISP entry protection

In addition to the three CRP modes, the user can prevent the sampling of pin the ISP entry pin for entering ISP mode and thereby release pin the ISP entry pin for other uses. This is called the NO_ISP mode. The NO_ISP mode can be entered by programming the pattern 0x4E69 7370 at location 0x0000 02FC.

27.4.2.2 Flash content protection mechanism

This part is equipped with the Error Correction Code (ECC) capable Flash memory. The purpose of an error correction module is twofold. Firstly, it decodes data words read from the memory into output data words. Secondly, it encodes data words to be written to the memory. The error correction capability consists of single bit error correction with Hamming code.

The operation of ECC is transparent to the running application. The ECC content itself is stored in a flash memory not accessible by user’s code to either read from it or write into it on its own. A byte of ECC corresponds to every consecutive 128 bits of the user accessible Flash. Consequently, Flash bytes from 0x0000 0000 to 0x0000 000F are protected by the first ECC byte, Flash bytes from 0x0000 0010 to 0x0000 001F are protected by the second ECC byte, etc.

Whenever the CPU requests a read from user’s Flash, both 128 bits of raw data containing the specified memory location and the matching ECC byte are evaluated. If the ECC mechanism detects a single error in the fetched data, a correction will be applied before data are provided to the CPU. When a write request into the user's Flash is made, write of user specified content is accompanied by a matching ECC value calculated and stored in the ECC memory.

When a sector of Flash memory is erased, the corresponding ECC bytes are also erased. Once an ECC byte is written, it can not be updated unless it is erased first. Therefore, for the implemented ECC mechanism to perform properly, data must be written into the flash memory in groups of 16 bytes (or multiples of 16), aligned as described above.

27.5 API description (ISP)

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 365. ISP command summary

ISP Command	Usage	Described in
Unlock	U <Unlock Code>	Table 366
Set Baud Rate	B <Baud Rate> <stop bit>	Table 367
Echo	A <setting>	Table 368
Write to RAM	W <start address> <number of bytes>	Table 369
Read Memory	R <address> <number of bytes>	Table 370
Prepare sector(s) for write operation	P <start sector number> <end sector number>	Table 371
Copy RAM to flash	C <Flash address> <RAM address> <number of bytes>	Table 372
Go	G <address> <Mode>	Table 373
Erase sector(s)	E <start sector number> <end sector number>	Table 374
Blank check sector(s)	I <start sector number> <end sector number>	Table 375
Read Part ID	J	Table 376
Read Boot code version	K	Table 378
Compare	M <address1> <address2> <number of bytes>	Table 379
ReadUID	N	Table 380

27.5.1 UART ISP Unlock

Table 366. ISP Unlock command

Command	U
Input	Unlock code: 23130_{10}

Table 366. ISP Unlock command

Command	U
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.

27.5.2 UART ISP Set Baud Rate

Table 367. 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.

27.5.3 UART ISP Echo

Table 368. 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.

27.5.4 UART ISP Write to RAM

The host should send the data only after receiving the CMD_SUCCESS return code. The host should send the check-sum after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes) i.e. it can hold 45 data bytes. When the data fits in less than 20 UU-encoded lines then the check-sum should be of the actual number of bytes sent. The ISP command handler compares it with the check-sum of the received bytes. If the check-sum matches, the ISP command handler responds with "OK<CR><LF>" to continue further transmission. If the check-sum does not match, the ISP command handler responds with "RESEND<CR><LF>". In response the host should retransmit the bytes.

Table 369. ISP Write to RAM command

Command	W
Input	**Start Address:** RAM address where data bytes are to be written. This address should be a word boundary. **Number of Bytes:** Number of bytes to be written. Count should be a multiple of 4
Return Code	CMD_SUCCESS | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to download data to RAM. Data should be in UU-encoded format. This command is blocked when code read protection is enabled.
Example	"W 268436224 4<CR><LF>" writes 4 bytes of data to address 0x1000 0300.

27.5.5 UART ISP Read Memory

The data stream is followed by the command success return code. The check-sum is sent after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes) i.e. it can hold 45 data bytes. When the data fits in less than 20 UU-encoded lines then the check-sum is of actual number of bytes sent. The host should compare it with the checksum of the received bytes. If the check-sum matches then the host should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match then the host should respond with "RESEND<CR><LF>". In response the ISP command handler sends the data again.

Table 370. ISP Read Memory command

Command	R
Input	**Start Address:** Address from where data bytes are to be read. This address should be a word boundary. **Number of Bytes:** Number of bytes to be read. Count should be a multiple of 4.
Return Code	CMD_SUCCESS followed by <actual data (UU-encoded)> | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not a multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to read data from RAM or flash memory. This command is blocked when code read protection is enabled.
Example	"R 268435456 4<CR><LF>" reads 4 bytes of data from address 0x1000 0000.

27.5.6 UART ISP Prepare sectors for write operation

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

Table 371. ISP Prepare sectors 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.

27.5.7 UART ISP Copy RAM to flash

When writing to the flash, the following limitations apply:

1. The smallest amount of data that can be written to flash by the copy RAM to flash command is 256 byte (equal to one page).
2. One page consists of 16 flash words (lines), and the smallest amount that can be modified per flash write is one flash word (one line). This limitation follows from the application of ECC to the flash write operation, see Section 27.4.2.2.
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 372. ISP Copy command

Command	C
Input	**Flash Address (DST):** Destination flash address where data bytes are to be written. The destination address should be a 256 byte boundary. **RAM Address(SRC):** Source RAM address from where data bytes are to be read. **Number of Bytes:** Number of bytes to be written. Should be 256 | 512 | 1024 | 4096.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not on word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR WRITE_OPERATION | BUSY | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to program the flash memory. The "Prepare Sectors for Write Operation" command should precede this command. The affected sectors are automatically protected again once the copy command is successfully executed. The boot block cannot be written by this command. This command is blocked when code read protection is enabled. Also see Section 27.4.2.2 for the number of bytes that can be written.
Example	"C 0 268467504 512<CR><LF>" copies 512 bytes from the RAM address 0x1000 0800 to the flash address 0.

27.5.8 UART ISP Go

The GO command is typically used after the flash image has been updated. After the update a reset is required. Therefore, the GO command should point to the RESET handler. Since the device is still in ISP mode, the RESET handler should do the following:

- Re-initialize the SP pointer to the application default.
- Set the SYSMEMREMAP to either 0x01 or 0x02.

While in ISP mode, the SYSMEMREMAP is set to 0x00.

Alternatively, the following snippet can be loaded into the RAM for execution:

```
SCB->AIRCR = 0x05FA0004; //issue system reset
while(1);                //should never come here
```

This snippet will issue a system reset request to the core.

The following ISP commands will send the system reset code loaded into 0x1000 000.

```
U 23130
W 268435456 16
0`4@"20%@_N<,[0#@!`#Z!0``
1462
G 268435456 T
```

Table 373. ISP Go command

Command	G
Input	**Address:** Flash or RAM address from which the code execution is to be started. This address should be on a word boundary. **Mode:** T (Execute program in Thumb Mode) | A (not allowed).
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.

27.5.9 UART ISP Erase sector

Table 374. 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.

27.5.10 UART ISP Blank check sector

Table 375. 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.

27.5.11 UART ISP Read Part Identification number

Table 376. ISP Read Part Identification command

Command	**J**
Input	None.
Return Code	CMD_SUCCESS followed by part identification number in ASCII (see Table 377 “Device identification numbers”).
Description	This command is used to read the part identification number.

Table 377. Device identification numbers

Device	Hex coding
LPC11U67JBD48	0x0000 BC88
LPC11U68JBD48	0x0000 7C08
LPC11U68JBD64	0x0000 7C08
LPC11U68JBD100	0x0000 7C00
LPC11E67JBD48	0x0000 BC81
LPC11E68JBD64	0x0000 7C01
LPC11E68JBD100	0x0000 7C01
LPC11U67JBD100	0x0000 BC80
LPC11U67JBD64	0x0000 BC88
LPC11U66JBD48	0x0000 DCC8
LPC11E68JBD48	0x0000 7C01
LPC11E67JBD100	0x0000 BC81
LPC11E67JBD64	0x0000 BC81
LPC11E66JBD48	0x0000 DCC1

27.5.12 UART ISP Read Boot code version number

Table 378. 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.

27.5.13 UART ISP Compare

Table 379. ISP Compare command

Command	M
Input	**Address1 (DST):** Starting flash or RAM address of data bytes to be compared. This address should be a word boundary. **Address2 (SRC):** Starting flash or RAM address of data bytes to be compared. This address should be a word boundary. **Number of Bytes:** Number of bytes to be compared; should be a multiple of 4.
Return Code	CMD_SUCCESS | (Source and destination data are equal) COMPARE_ERROR | (Followed by the offset of first mismatch) COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED | PARAM_ERROR |
Description	This command is used to compare the memory contents at two locations. **Compare result may not be correct when source or destination address contains any of the first 512 bytes starting from address zero. First 512 bytes are re-mapped to boot ROM**
Example	"M 8192 268468224 4<CR><LF>" compares 4 bytes from the RAM address 0x1000 8000 to the 4 bytes from the flash address 0x2000.

27.5.14 UART ISP ReadUID

Table 380. ReadUID command

Command	N
Input	None
Return Code	CMD_SUCCESS followed by four 32-bit words of a unique serial number in ASCII format. The word sent at the lowest address is sent first.
Description	This command is used to read the unique ID.

27.5.15 ISP Return Codes

Table 381. ISP Return Codes Summary

Return Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable.
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid or end sector number is greater than start sector number.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data not equal.
11	BUSY	Flash programming hardware interface is busy.
12	PARAM_ERROR	Insufficient number of parameters or invalid parameter.
13	ADDR_ERROR	Address is not on word boundary.
14	ADDR_NOT_MAPPED	Address is not mapped in the memory map. Count value is taken in to consideration where applicable.
15	CMD_LOCKED	Command is locked.
16	INVALID_CODE	Unlock code is invalid.
17	INVALID_BAUD_RATE	Invalid baud rate setting.
18	INVALID_STOP_BIT	Invalid stop bit setting.
19	CODE_READ_PROTECTION_ENABLED	Code read protection enabled.

27.6 API description (IAP)

Remark: When using the IAP commands, configure the power profiles in Default mode. See Section 28.5.2 “set_power”.

For in application programming the IAP routine should be called with a word pointer in register r0 pointing to memory (RAM) containing command code and parameters. The result of the IAP command is returned in the result table pointed to by register r1. The user can reuse the command table for result by passing the same pointer in registers r0 and r1. The parameter table should be big enough to hold all the results in case the number of results are more than number of parameters. Parameter passing is illustrated in the Figure 88.

The number of parameters and results vary according to the IAP command. The maximum number of parameters is 5, passed to the "Copy RAM to FLASH" command. The maximum number of results is 5, returned by the "ReadUID" command. The command handler sends the status code INVALID_COMMAND when an undefined command is received. The IAP routine resides at 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 int command_param[5];
unsigned int status_result[5];
```

or

```
unsigned int * command_param;
unsigned int * status_result;
command_param = (unsigned int *) 0x...
status_result =(unsigned int *) 0x...
```

Define pointer to function type, which takes two parameters and returns void. Note the IAP returns the result with the base address of the table residing in R1.

```
typedef void (*IAP)(unsigned int [],unsigned int[]);
IAP iap_entry;
```

Setting the function pointer:

```
#define IAP_LOCATION 0x1fff1ff1
```

```
iap_entry=(IAP) IAP_LOCATION;
```

To call the IAP, use the following statement.

```
iap_entry (command_param,status_result);
```

Up to 4 parameters can be passed in the r0, r1, r2 and r3 registers respectively (see the *ARM Thumb Procedure Call Standard SWS ESPC 0002 A-05)*. Additional parameters are passed on the stack. Up to 4 parameters can be returned in the r0, r1, r2 and r3 registers respectively. Additional parameters are returned indirectly via memory. 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 382. IAP Command Summary

IAP Command	Command Code	Described in
Prepare sector(s) for write operation	50 (decimal)	Table 383
Copy RAM to flash	51 (decimal)	Table 384
Erase sector(s)	52 (decimal)	Table 385
Blank check sector(s)	53 (decimal)	Table 386
Read Part ID	54 (decimal)	Table 387
Read Boot code version	55 (decimal)	Table 388
Compare	56 (decimal)	Table 389
Reinvoke ISP	57 (decimal)	Table 390
Read UID	58 (decimal)	Table 391
Erase page	59 (decimal)	Table 392
EEPROM Write	61(decimal)	Table 393
EEPROM Read	62(decimal)	Table 394

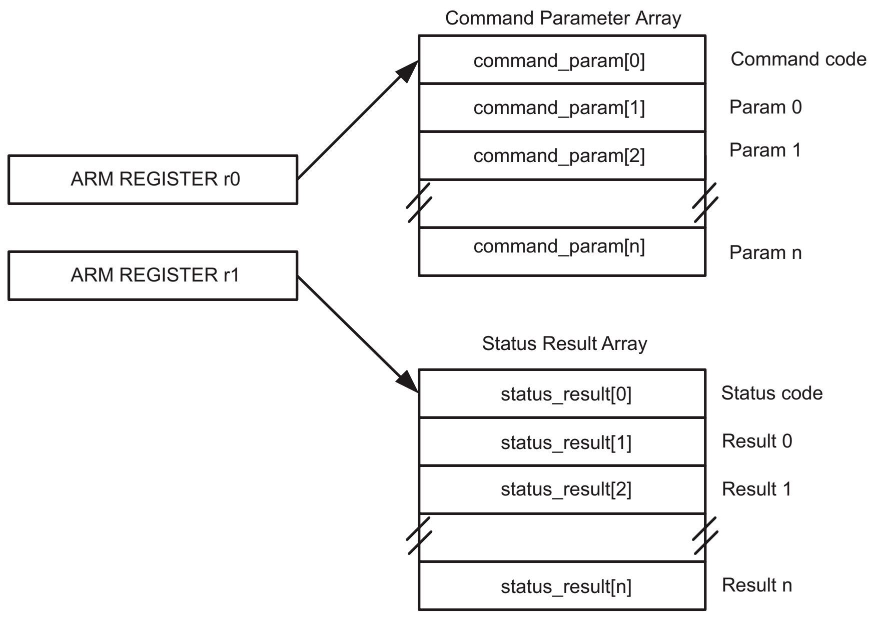


Fig 88. IAP parameter passing

27.6.1 IAP Prepare sector for write operation

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

Table 383. IAP Prepare sector for write operation command

Command	Prepare sector for write operation
Input	**Command code: 50 (decimal)** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number).
Status code	CMD_SUCCESS | BUSY | INVALID_SECTOR
Result	None
Description	This command must be executed before executing "Copy RAM to flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. The boot sector can not be prepared by this command. To prepare a single sector use the same "Start" and "End" sector numbers.

27.6.2 IAP Copy RAM to flash

See Section 27.5.7 for limitations on the write-to-flash process.

Table 384. IAP Copy RAM to flash command

Command	Copy RAM to flash
Input	**Command code: 51 (decimal)** **Param0(DST):** Destination flash address where data bytes are to be written. This address should be a 256 byte boundary. **Param1(SRC):** Source RAM address from which data bytes are to be read. This address should be a word boundary. **Param2:** Number of bytes to be written. Should be 256 | 512 | 1024 | 4096. **Param3:** System Clock Frequency (CCLK) in kHz.
Status code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not a word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | BUSY
Result	None
Description	This command is used to program the flash memory. The affected sectors should be prepared first by calling "Prepare Sector for Write Operation" command. The affected sectors are automatically protected again once the copy command is successfully executed. The boot sector can not be written by this command. Also see Section 27.4.2.2 for the number of bytes that can be written.

27.6.3 IAP Erase Sector

Table 385. IAP Erase Sector 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.
Status code	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
Result	None
Description	This command is used to erase a sector or multiple sectors of on-chip flash memory. The boot sector can not be erased by this command. To erase a single sector use the same "Start" and "End" sector numbers.

27.6.4 IAP Blank check sector

Table 386. IAP Blank check sector command

Command	Blank check sector(s)
Input	**Command code: 53 (decimal)** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number).
Status code	CMD_SUCCESS | BUSY | SECTOR_NOT_BLANK | INVALID_SECTOR
Result	**Result0:** Offset of the first non blank word location if the Status Code is SECTOR_NOT_BLANK. **Result1:** Contents of non blank word location.
Description	This command is used to blank check a sector or multiple sectors of on-chip flash memory. To blank check a single sector use the same "Start" and "End" sector numbers.

27.6.5 IAP Read Part Identification number

Table 387. IAP Read Part Identification command

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

27.6.6 IAP Read Boot code version number

Table 388. IAP Read Boot Code version number command

Command	Read boot code version number
Input	**Command code: 55 (decimal)** **Parameters:** None
Status code	CMD_SUCCESS
Result	**Result0:** Boot code version number. Read as <byte1(Major)>.<byte0(Minor)>
Description	This command is used to read the boot code version number.

27.6.7 IAP Compare

Table 389. IAP Compare command

Command	Compare
Input	**Command code: 56 (decimal)** **Param0(DST):** Starting flash or RAM address of data bytes to be compared. This address should be a word boundary. **Param1(SRC):** Starting flash or RAM address of data bytes to be compared. This address should be a word boundary. **Param2:** Number of bytes to be compared; should be a multiple of 4.
Status code	CMD_SUCCESS | COMPARE_ERROR | COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED
Result	**Result0:** Offset of the first mismatch if the Status Code is COMPARE_ERROR.
Description	This command is used to compare the memory contents at two locations. **The result may not be correct when the source or destination includes any of the first 512 bytes starting from address zero. The first 512 bytes can be re-mapped to RAM.**

27.6.8 IAP Reinvoke ISP

Table 390. Reinvoke ISP

Command	Compare
Input	**Command code: 57 (decimal)**
Status code	None
Result	**None.**
Description	This command is used to invoke the bootloader in ISP mode. It maps boot vectors, sets PCLK = CCLK, configures UART pins RXD and TXD, resets counter/timer CT32B1 and resets the USART0 FDR (see Table 167). This command may be used when a valid user program is present in the internal flash memory and the ISP entry pin is not accessible to force the ISP mode.

27.6.9 IAP ReadUID

Table 391. IAP ReadUID command

Command	Compare
Input	**Command code: 58 (decimal)**
Status code	CMD_SUCCESS
Result	**Result0:** The first 32-bit word (at the lowest address). **Result1:** The second 32-bit word. **Result2:** The third 32-bit word. **Result3:** The fourth 32-bit word.
Description	This command is used to read the unique ID.

27.6.10 IAP Erase page

Table 392. 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.
Status code	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
Result	None
Description	This command is used to erase a page or multiple pages of on-chip flash memory. To erase a single page use the same "start" and "end" page numbers.

27.6.11 IAP Write EEPROM

Table 393. IAP Write EEPROM command

Command	Compare
Input	**Command code: 61 (decimal)** **Param0:** EEPROM address. **Param1:** RAM address. **Param2:** Number of bytes to be written. **Param3:** System Clock Frequency (CCLK) in kHz.
Status code	CMD_SUCCESS | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED
Result	None
Description	Data is copied from the RAM address to the EEPROM address. **Remark:** The top 64 bytes of the 4 KB EEPROM memory are reserved and cannot be written to. The entire EEPROM is writable for smaller EEPROM sizes.

27.6.12 IAP Read EEPROM

Table 394. IAP Read EEPROM command

Command	Compare
Input	**Command code: 62 (decimal)** **Param0:** EEPROM address. **Param1:** RAM address. **Param2:** Number of bytes to be read. **Param3:** System Clock Frequency (CCLK) in kHz.
Status code	CMD_SUCCESS | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED
Result	None
Description	Data is copied from the EEPROM address to the RAM address.

27.6.13 IAP Status codes

Table 395. IAP Status codes Summary

Status Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on a word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable.
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data is not same.
11	BUSY	flash programming hardware interface is busy.

27.7 Functional description

27.7.1 ISP/IAP communication protocol

All ISP commands should be sent as single ASCII strings. Strings should be terminated with Carriage Return (CR) and/or Line Feed (LF) control characters. Extra <CR> and <LF> characters are ignored. All ISP responses are sent as <CR><LF> terminated ASCII strings. Data is sent and received in UU-encoded format.

27.7.1.1 ISP command format

"Command Parameter_0 Parameter_1 ... Parameter_n<CR><LF>" "Data" (Data only for Write commands).

27.7.1.2 ISP response format

"Return_Code<CR><LF>Response_0<CR><LF>Response_1<CR><LF>... Response_n<CR><LF>" "Data" (Data only for Read commands).

27.7.1.3 ISP data format

The data stream is in UU-encoded format. The UU-encode algorithm converts 3 bytes of binary data in to 4 bytes of printable ASCII character set. It is more efficient than Hex format which converts 1 byte of binary data in to 2 bytes of ASCII hex. The sender should send the check-sum after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes) i.e. it can hold 45 data bytes. The receiver should compare it with the check-sum of the received bytes. If the check-sum matches then the receiver should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match the receiver should respond with "RESEND<CR><LF>". In response the sender should retransmit the bytes.

27.7.1.4 ISP flow control

A software XON/XOFF flow control scheme is used to prevent data loss due to buffer overrun. When the data arrives rapidly, the ASCII control character DC3 (stop) is sent to stop the flow of data. Data flow is resumed by sending the ASCII control character DC1 (start). The host should also support the same flow control scheme.

27.7.1.5 ISP command abort

Commands can be aborted by sending the ASCII control character "ESC". This feature is not documented as a command under "ISP Commands" section. Once the escape code is received the ISP command handler waits for a new command.

27.7.1.6 Interrupts during ISP

The boot block interrupt vectors located in the boot block of the flash are active after any reset.

27.7.1.7 Interrupts during IAP

The on-chip flash memory and EEPORM are 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.

27.7.1.8 RAM used by ISP command handler

ISP commands use on-chip RAM from 0x1000 017C to 0x1000 025B. The user could use this area, but the contents may be lost upon reset. Flash programming commands use the top 32 bytes of on-chip RAM. The stack is located at RAM top – 32 bytes. The maximum stack usage is 256 bytes and grows downwards.

27.7.1.9 RAM used by IAP command handler

Flash programming commands use the top 32 bytes of on-chip RAM. The maximum stack usage in the user allocated stack space is 128 bytes and grows downwards.

27.7.2 UART communication protocol

If the UART is selected, the host should send a ’?’ (0x3F) as a synchronization character and wait for a response. The host side serial port settings should be 8 data bits, 1 stop bit and no parity. The auto-baud routine measures the bit time of the received synchronization character in terms of its own frequency and programs the baud rate generator of the serial port. It also sends an ASCII string ("Synchronized<CR><LF>") to the host. In response to this host should send the same string ("Synchronized<CR><LF>"). The auto-baud routine looks at the received characters to verify synchronization. If synchronization is verified then "OK<CR><LF>" string is sent to the host. Host should respond by sending the crystal frequency (in kHz) at which the part is running. For example, if the part is running at 10 MHz, the response from the host should be "10000<CR><LF>". "OK<CR><LF>" string is sent to the host after receiving the crystal frequency. If synchronization is not verified then the auto-baud routine waits again for a synchronization character. For auto-baud to work correctly in case of user invoked ISP, the CCLK frequency should be greater than or equal to 10 MHz. In USART ISP mode, the part is clocked by the IRC and the crystal frequency is ignored.

Once the crystal frequency is received the part is initialized and the ISP command handler is invoked. For safety reasons an "Unlock" command is required before executing the commands resulting in flash erase/write operations and the "Go" command. The rest of the commands can be executed without the unlock command. The Unlock command is required to be executed once per ISP session. The Unlock command is explained in Section 27.5.1.

27.7.3 USB communication protocol

This part is enumerated as a Mass Storage Class (MSC) device to a PC or another embedded system. In order to connect via the USB interface, the part must use the external crystal at a frequency of 12 MHz. The MSC device presents an easy integration with the PC’s operating system. The part’s flash memory space is represented as a drive in the host file system. The entire available user flash is mapped to a file of the size of the part’s flash in the host’s folder with the default name ‘firmware.bin’. The ‘firmware.bin’ file can be deleted and a new file can be copied into the directory, thereby updating the user code in flash. Note that the filename of the new flash image file is not important. After a reset or a power cycle, the new file is visible in the host’s file system under it’s default name ‘firmware.bin’.

The code read protection (CRP, see Table 396) level determines how the flash is reprogrammed:

If CRP1 or CRP2 is enabled, the user flash is erased when the file is deleted.

If CRP1 is enabled or no CRP is selected, the user flash is erased and reprogrammed when the new file is copied. However, only the area occupied by the new file is erased and reprogrammed.

Remark: The only commands supported for the part’s flash image folder are copy and delete.

Three Code Read Protection (CRP) levels can be enabled for flash images updated through USB (see Section 27.7.3 for details). The volume label on the MSCD indicates the CRP status.

Table 396. CRP levels for USB boot images

CRP status	Volume label	Description
No CRP	CRP DISABLD	The user flash can be read or written.
CRP1	CRP1 ENABLD	The user flash content cannot be read but can be updated. The flash memory sectors are updated depending on the new firmware image.
CRP2	CRP2 ENABLD	The user flash content cannot be read but can be updated. The entire user flash memory is erased before writing the new firmware image.
CRP3	CRP3 ENABLD	The user flash content cannot be read or updated. The bootloader always executes the user application if valid.

27.7.3.1 Usage note

When programming flash images via Flash Magic or Serial Wire Debugger (SWD), the user code valid signature is automatically inserted by the programming utility. When using USB ISP, the user code valid signature must be either part of the vector table, or the axf or binary file must be post-processed to insert the checksum.

UM10732

Chapter 28: LPC11U6x/E6x Power profiles/Power control API

Rev. 1.8 — 19 December 2016 **User manual**

28.1 How to read this chapter

The power profiles are available for all parts.

28.2 Features

- ROM-based application.
- Simple API to control power consumption and wake-up in all power modes.
- Manage power consumption for sleep and active modes.
- Configure PLL.

28.3 Basic configuration

Specific power profile settings are required in the following situations:

- When using the USB, configure the power profiles in Default mode.
- When using IAP commands, configure the power profiles in Default mode.

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.

28.4 General description

The power consumption in Active and Sleep modes can be optimized for the application through simple calls to the power profile. The power configuration routine configures the part 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.

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 89 shows the pointer structure used to call the Power Profiles API.

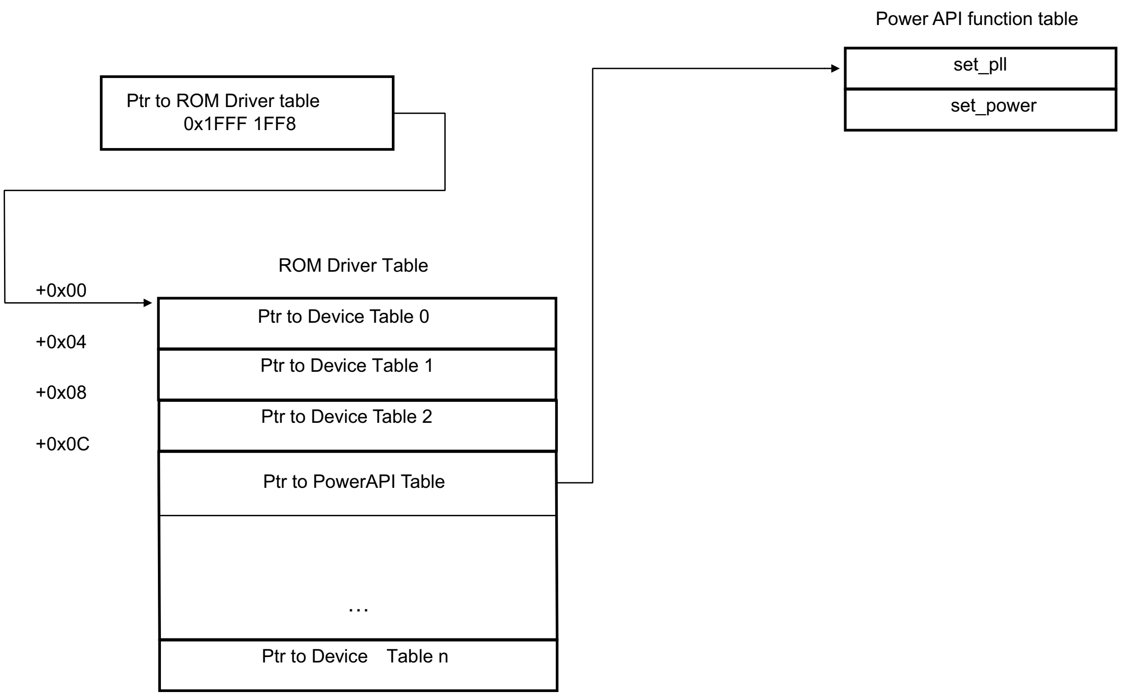


Fig 89. Power profiles pointer structure

28.5 API description

The power profile API provides functions to configure the system clock and optimize the system setting for lowest power consumption.

Table 397. Power profile API calls

API call	Description	Reference
`void set_pll(command, result);`	Power API set_pll routine for active and sleep modes	Table 398
`void set_power(command, result);`	Power API set_power routine for active and sleep modes	Table 399

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

```
typedef struct PWRD_API {
        void (*set_pll)(uint32_t cmd[], uint32_t resp[]); /*!< Set PLL function */
        void (*set_pll)(unsigned int cmd[], unsigned int resp[]); /*!< Set power function */
} PWRD_API_T;

#define LPC_PWRD_API ((LPC_ROM_API)->pPWRD)
```

The ROM API table shown in Section 26.5.5 “ROM-based APIs” must be included in the code.

28.5.1 set_pll

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

Remark: Before this routine is invoked, the PLL clock source (IRC/system oscillator) must be selected (Table 33), the main clock source must be set to the input clock to the system PLL, and the system/AHB clock divider must be set to 1 (Table 39).

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.

Table 398. set_pll routine

Routine	set_pll
Prototype	`void set_pll(command, result);`
Input parameter	**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)
Return	None.
Description	Sets the system PLL.

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
```

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

set_pll configures a setup in which the main clock does not exceed 50 MHz. It easily finds a solution when the ratio between the expected system clock and the system PLL input frequency is an integer value, but it can also find solutions in other cases.

The system PLL input frequency (Param0) must be between 10000 to 25000 kHz (10 MHz to 25 MHz) inclusive. The expected system clock (Param1) must be between 1 and 50000 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.

28.5.1.2 Param2: mode

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

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

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

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

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

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

28.5.1.3 Param3: system PLL lock time-out

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

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

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

28.5.2 set_power

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

Remark: Use the set_power routine with SYSAHBCLKDIV = 1 (System clock divider register, see Table 39).

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

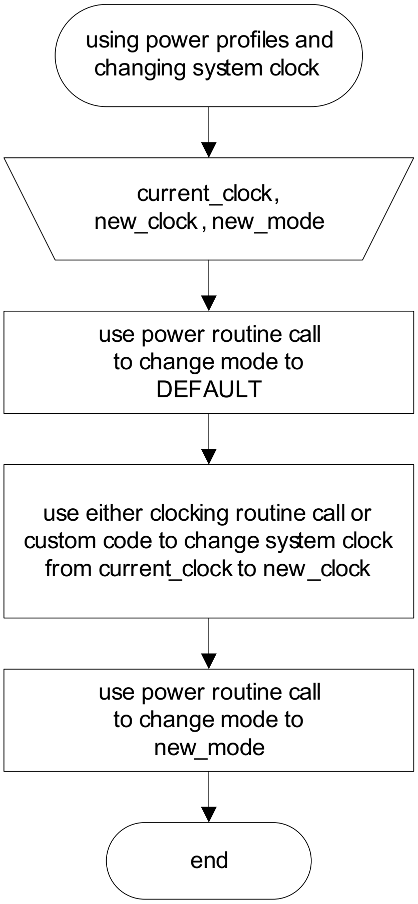


Fig 90. Power profiles usage

Table 399. set_power routine

Routine	set_power
Prototype	`void set_power(command, result);`
Input parameter	**Param0:** main clock (in MHz) **Param1:** mode (PWR_DEFAULT, PWR_CPU_PERFORMANCE, PWR_EFFICIENCY, PWR_LOW_CURRENT) **Param2:** system clock (in MHz)

Table 399. set_power routine *…continued*

Routine	**set_power**
Result	**Result0:** PWR_CMD_SUCCESS | PWR_INVALID_FREQ | PWR_INVALID_MODE
Return	None.
Description	Configures the power mode in active and sleep modes.

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
```

28.5.2.1 Param0: main clock

The main clock is the clock rate the microcontroller uses to source the system’s and the peripherals’ clock. It is configured by either a successful execution of the clocking routine call or a similar code provided by the user. This operand must be an integer between 1 to 50 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.

28.5.2.2 Param1: mode

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

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

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

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

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

28.5.2.3 Param2: system clock

The system clock is the clock rate at which the microcontroller core is running when set_power is called. This parameter is an integer between from 1 and 50 MHz inclusive.

28.5.3 Error codes

The error code is returned in the result field of the set_pll and set_power API functions.

Table 400. Error codes for set_pll

Return code	Error Code	Description
0	PLL_CMD_SUCCESS	-
1	PLL_INVALID_FREQ	-
2	PLL_INVALID_MODE	-
3	PLL_FREQ_NOT_FOUND	-
4	PLL_NOT_LOCKED	-

```
#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
```

Table 401. Error codes for set_power

Return code	Error Code	Description
0	PARAM_CMD_SUCCESS	-
1	PARAM_INVALID_FREQ	-
2	PARAM_INVALID_MODE	-

```
#define        PARAM_CMD_SUCCESS       0
#define        PARAM_INVALID_FREQ      1
#define        PARAM_INVALID_MODE      2
```

28.6 Functional description

28.6.1 Clock control

See Section 28.6.1.1 to Section 28.6.1.6 for examples of the clock control API.

28.6.1.1 Invalid frequency (device maximum clock rate exceeded)

```
command[0] = 12000;
command[1] = 840000;
command[2] = CPU_FREQ_EQU;
command[3] = 0;
LPC_PWRD_API->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock and a system clock of exactly 84 MHz. The application was ready to infinitely wait for the PLL to lock. But the expected system clock of 84 MHz exceeds the maximum of 50 MHz. Therefore set_pll returns PLL_INVALID_FREQ in result[0] and 12000 in result[1] without changing the PLL settings.

28.6.1.2 Invalid frequency selection (system clock divider restrictions)

```
command[0] = 12000;
command[1] = 40;
command[2] = CPU_FREQ_LTE;
command[3] = 0;
```

```
LPC_PWRD_API->set_pll(command, result);
```

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

28.6.1.3 Exact solution cannot be found (PLL)

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

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

28.6.1.4 System clock less than or equal to the expected value

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

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

28.6.1.5 System clock greater than or equal to the expected value

```
command[0] = 12000;
command[1] = 20000;
command[2] = CPU_FREQ_GTE;
command[3] = 0;
LPC_PWRD_API->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock, a system clock of at least 20 MHz and no locking time-out. set_pll returns PLL_CMD_SUCCESS in result[0] and 24000 in result[1]. The new system clock is 24 MHz.

28.6.1.6 System clock approximately equal to the expected value

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

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

28.6.2 Power control

See Section 28.6.1.1 and Section 28.6.2.2 for examples of the power control API.

28.6.2.1 Invalid frequency (device maximum clock rate exceeded)

```
command[0] = 50;
command[1] = PWR_CPU_PERFORMANCE;
command[2] = 84;
LPC_PWRD_API->set_power(command, result);
```

The above setup would be used in a system running at the main and system clock of 30 MHz, with a need for maximum CPU processing power. Since the specified 84 MHz clock is above the 50 MHz maximum, set_power returns PWR_INVALID_FREQ in result[0] without changing anything in the existing power setup.

28.6.2.2 An applicable power setup

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

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

UM10732

Chapter 29: LPC11U6x/E6x Integer division routines

Rev. 1.8 — 19 December 2016 **User manual**

29.1 How to read this chapter

The ROM-based 32-bit integer division routines are available on all parts.

29.2 Features

- Performance-optimized signed/unsigned integer division.
- Performance-optimized signed/unsigned integer division with remainder.
- ROM-based routines to reduce code size.
- Support for integers up to 32 bit.
- ROM calls can easily be added to EABI-compliant functions to overload “/” and “%” operators in C.

29.3 General description

The API calls to the ROM are performed by executing functions which are pointed by a pointer within the ROM Driver Table. Figure 91 shows the pointer structure used to call the Integer divider API.

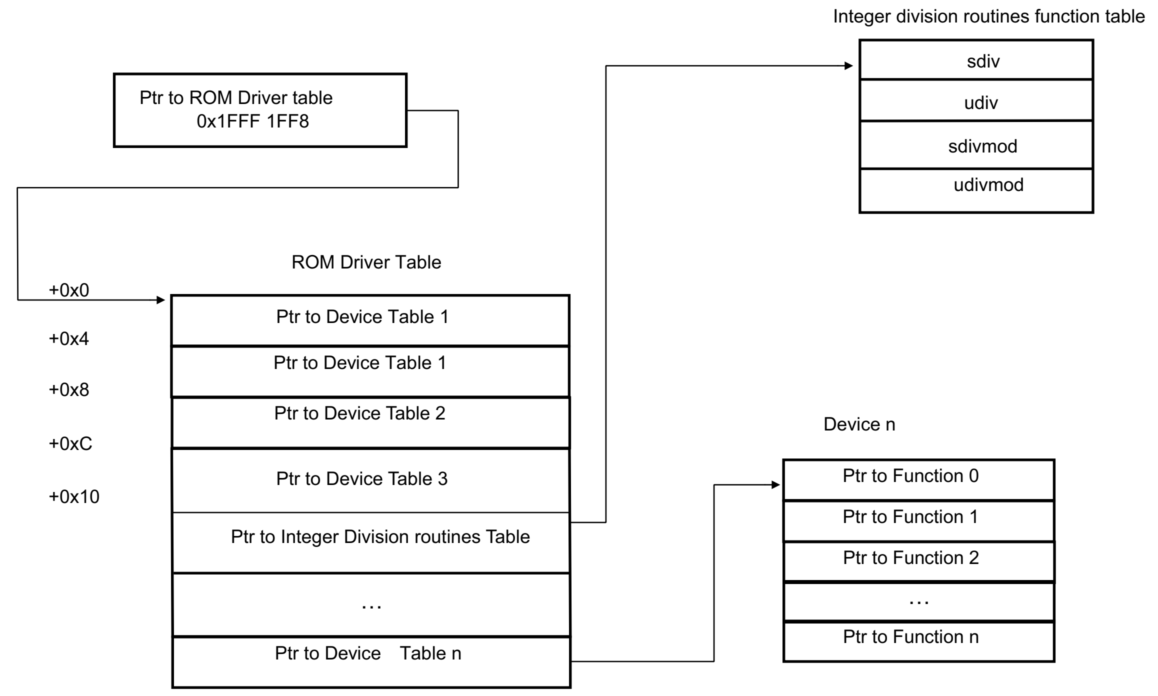


Fig 91. ROM pointer structure

29.4 API description

The integer division routines perform arithmetic integer division operations and can be called in the application code through simple API calls.

Table 402. Divide API calls

API call	Description	Reference
`int(*sdiv)(int numerator, int denominator);`	Signed integer division	Table 403
`unsigned(*udiv) (int numerator, int denominator);`	Unsigned integer division	Table 404
`sdiv_t (*sdivmod)(int numerator, int denominator);`	Signed integer division with remainder	Table 405
`udiv_t (*udivmod)(unsigned numerator, unsigned denominator);`	Unsigned integer division with remainder	Table 406

The following function prototypes are used:

```
typedef struct {
        int quot;                /*!< Quotient */
        int rem;                 /*!< Remainder */
} IDIV_RETURN_T;

typedef struct {
        unsigned quot; /*!< Quotient */
        unsigned rem;  /*!< Reminder */
} UIDIV_RETURN_T;

typedef struct {
        int (*sidiv)(int numerator, int denominator);           /*!< Signed integer division */
        unsigned (*uidiv)(unsigned numerator, unsigned denominator); /*!< Unsigned integer division */
        IDIV_RETURN_T (*sidivmod)(int numerator, int denominator); /*!< Signed integer division with remainder */
        UIDIV_RETURN_T (*uidivmod)(unsigned numerator, unsigned denominator);/*!< Unsigned integer division
                with remainder */
} ROM_DIV_API_T;


ROM_DIV_API_T const *pROMDiv = LPC_ROM_API->divApiBase;
```

The ROM API table shown in Section 26.5.5 “ROM-based APIs” must be included in the code.

29.4.1 DIV signed integer division

Table 403. sidiv

Routine	sidiv
Prototype	`int(*sidiv)(int32_t numerator, int32_t denominator);`
Input parameter	numerator: Numerator signed integer. denominator: Denominator signed integer.
Return	Signed division result without remainder.
Description	Signed integer division

29.4.2 DIV unsigned integer division

Table 404. uidiv

Routine	uidiv
Prototype	`int(*uidiv)(int32_t numerator, int32_t denominator);`
Input parameter	numerator: Numerator signed integer. denominator: Denominator signed integer.
Return	Unsigned division result without remainder.
Description	Unsigned integer division

29.4.3 DIV signed integer division with remainder

Table 405. sidivmod

Routine	sidivmod
Prototype	`IDIV_RETURN_T (*sidivmod) (int32_t numerator, int32_t denominator);`
Input parameter	numerator: Numerator signed integer. denominator: Denominator signed integer.
Return	Signed division result remainder.
Description	Signed integer division with remainder

29.4.4 DIV unsigned integer division with remainder

Table 406. uidivmod

Routine	uidivmod
Prototype	`UIDIV_RETURN_T(*uidiv)(uint32_t numerator, uint32_t denominator);`
Input parameter	numerator: Numerator unsigned integer. denominator: Denominator unsigned integer.
Return	Unsigned division result with remainder.
Description	Unsigned integer division

29.5 Functional description

29.5.1 Signed division

The example C-code listing below shows how to perform a signed integer division via the ROM API.

```
/* Divide (-99) by (+6) */
int32_t result;
result = pROMDiv->sidiv(-99, 6);
/* result now contains (-16) */
```

29.5.2 Unsigned division with remainder

The example C-code listing below shows how to perform an unsigned integer division with remainder via the ROM API.

```
/* Modulus Divide (+99) by (+4) */
uidiv_return result;
result = pROMDiv-> uidivmod (+99, 4);
/* result.div contains (+24) */
/* result.mod contains (+3) */
```

UM10732

Chapter 30: LPC11U6x/E6x I2C-bus ROM API

Rev. 1.8 — 19 December 2016 **User manual**

30.1 How to read this chapter

The I2C-bus ROM API is available on all parts.

30.2 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.

30.3 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.

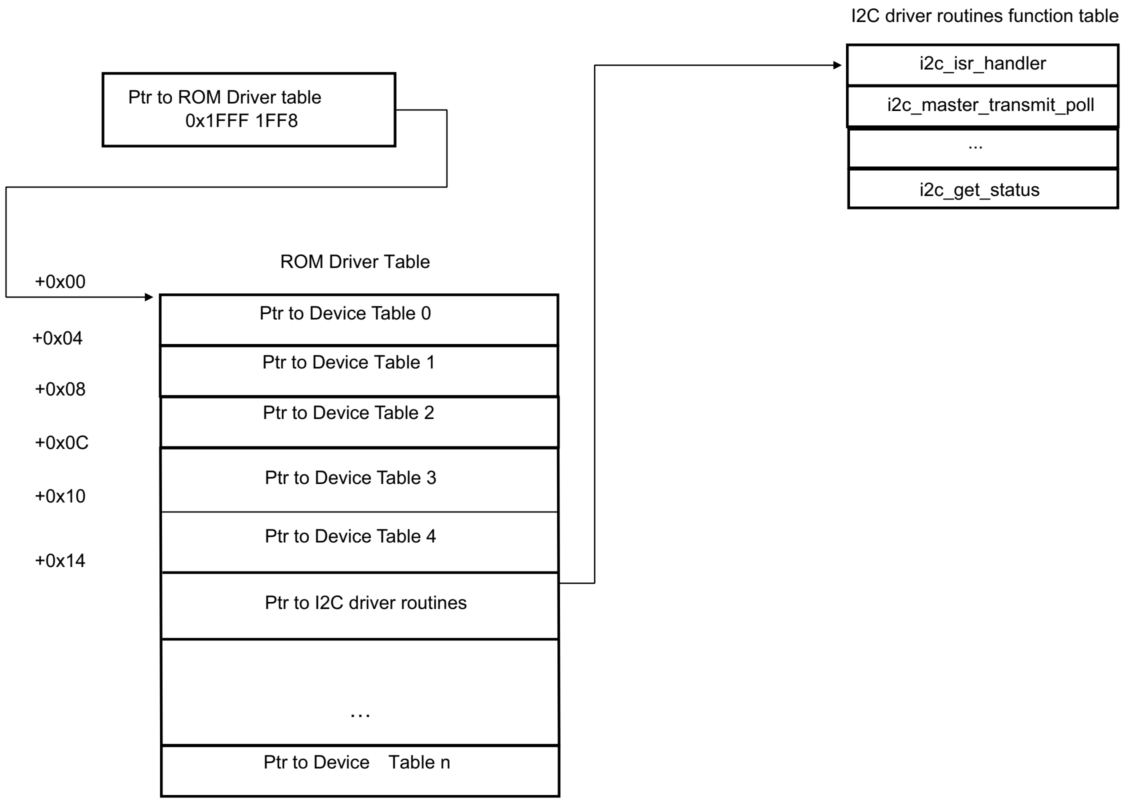


Fig 92. I2C-bus driver routines pointer structure

30.4 API description

The I2C API contains functions to configure the I2C and send and receive data in master and slave modes.

Table 407. I2C API calls

API call	Description	Reference
void i2c_isr_handler(I2C_HANDLE_T*);	I2C ROM Driver interrupt service routine.	Table 408
Master functions		
ErrorCode_t i2c_master_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*);	I2C Master Transmit Polling	Table 409
ErrorCode_t i2c_master_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);	I2C Master Receive Polling	Table 410
ErrorCode_t i2c_master_tx_rx_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);	I2C Master Transmit and Receive Polling	Table 411
ErrorCode_t i2c_master_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);	I2C Master Transmit Interrupt	Table 412
ErrorCode_t i2c_master_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);	I2C Master Receive Interrupt	Table 413
ErrorCode_t i2c_master_tx_rx_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);	I2C Master Transmit Receive Interrupt	Table 414
Slave functions		
ErrorCode_t i2c_slave_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);	I2C Slave Receive Polling	Table 415

Table 407. I2C API calls

API call	Description	Reference
ErrorCode_t i2c_slave_transmit_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);	I2C Slave Transmit Polling	Table 416
ErrorCode_t i2c_slave_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);	I2C Slave Receive Interrupt	Table 417
ErrorCode_t i2c_slave_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);	I2C Slave Transmit Interrupt	Table 418
ErrorCode_t i2c_set_slave_addr(I2C_HANDLE_T*, slave_addr_0_3, slave_mask_0_3);	I2C Set Slave Address	Table 419
Set-up functions		
uint32_t i2c_get_mem_size(void)	I2C Get Memory Size	Table 420
I2C_HANDLE_T* i2c_setup(i2c_base_addr, *start_of_ram);	I2C Setup	Table 421
ErrorCode_t i2c_set_bitrate(I2C_HANDLE_T*, P_clk_in_hz, bitrate_in_bps);	I2C Set Bit Rate	Table 422
uint32_t i2c_get_firmware_version(void);	I2C Get Firmware Version	Table 423
I2C_MODE_T i2c_get_status(I2C_HANDLE_T*);	I2C Get Status	Table 424

The following structure has to be defined to use the I2C API:

```
typedef struct  I2CD_API {
        /*!< Interrupt Support Routine */
        void (*i2c_isr_handler)(I2C_HANDLE_T *handle);
        /*!< MASTER functions */
        ErrorCode_t (*i2c_master_transmit_poll)(I2C_HANDLE_T *handle, I2C_PARAM_T *param, I2C_RESULT_T
         *result);
        ErrorCode_t (*i2c_master_receive_poll)(I2C_HANDLE_T *handle, I2C_PARAM_T *param, I2C_RESULT_T
         *result);
        ErrorCode_t (*i2c_master_tx_rx_poll)(I2C_HANDLE_T *handle, I2C_PARAM_T *param, I2C_RESULT_T
         *result);
        ErrorCode_t (*i2c_master_transmit_intr)(I2C_HANDLE_T *handle, I2C_PARAM_T *param, I2C_RESULT_T
         *result);
        ErrorCode_t (*i2c_master_receive_intr)(I2C_HANDLE_T *handle, I2C_PARAM_T *param, I2C_RESULT_T
         *result);
        ErrorCode_t (*i2c_master_tx_rx_intr)(I2C_HANDLE_T *handle, I2C_PARAM_T *param, I2C_RESULT_T
         *result);
        /*!< SLAVE functions */
        ErrorCode_t (*i2c_slave_receive_poll)(I2C_HANDLE_T *handle, I2C_PARAM_T *param, I2C_RESULT_T
         *result);
        ErrorCode_t (*i2c_slave_transmit_poll)(I2C_HANDLE_T *handle, I2C_PARAM_T *param, I2C_RESULT_T
        *result);
        ErrorCode_t (*i2c_slave_receive_intr)(I2C_HANDLE_T *handle, I2C_PARAM_T *param, I2C_RESULT_T
         *result);
        ErrorCode_t (*i2c_slave_transmit_intr)(I2C_HANDLE_T *handle, I2C_PARAM_T *param, I2C_RESULT_T
         *result);
        ErrorCode_t (*i2c_set_slave_addr)(I2C_HANDLE_T *handle, uint32_t slave_addr_0_3, uint32_t
         slave_mask_0_3);
        /*!< OTHER support functions */
        uint32_t        (*i2c_get_mem_size)(void);
        I2C_HANDLE_T *  (*i2c_setup)( uint32_t  i2c_base_addr, uint32_t * start_of_ram);
        ErrorCode_t     (*i2c_set_bitrate)(I2C_HANDLE_T *handle, uint32_t  p_clk_in_hz, uint32_t bitrate_in_bps);
        uint32_t        (*i2c_get_firmware_version)(void);
        CHIP_I2C_MODE_T (*i2c_get_status)(I2C_HANDLE_T *handle);
} I2CD_API_T;
```

```
#define LPC_I2CD_API((LPC_ROM_API)->pI2CD)
```

The ROM API table shown in Section 26.5.5 “ROM-based APIs” must be included in the code.

30.4.1 ISR handler

Table 408. ISR handler

Routine	ISR handler
Prototype	`void i2c_isr_handler(I2C_HANDLE_T*);`
Input parameter	I2C_HANDLE_T:Handle to the I2C instance.
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.

30.4.2 I2C Master Transmit Polling

Table 409. 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 I2C instance. 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.

30.4.3 I2C Master Receive Polling

Table 410. 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 I2C instance. 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.

30.4.4 I2C Master Transmit and Receive Polling

Table 411. 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 I2C instance. 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.

30.4.5 I2C Master Transmit Interrupt

Table 412. 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 I2C instance. 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.

30.4.6 I2C Master Receive Interrupt

Table 413. 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 I2C instance. 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.

30.4.7 I2C Master Transmit Receive Interrupt

Table 414. 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 I2C instance. 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.

30.4.8 I2C Slave Receive Polling

Table 415. 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 I2C instance. 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.

30.4.9 I2C Slave Transmit Polling

Table 416. 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 I2C instance. 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.

30.4.10 I2C Slave Receive Interrupt

Table 417. 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 I2C instance. 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.

30.4.11 I2C Slave Transmit Interrupt

Table 418. 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 I2C instance. 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.

30.4.12 I2C Set Slave Address

Table 419. 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 I2C instance. 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.

30.4.13 I2C Get Memory Size

Table 420. I2C Get Memory Size

Routine	I2C Get Memory Size
Prototype	uint32_t i2c_get_mem_size(void);
Input parameter	None.
Return	uint32.
Description	Returns the number of bytes in SRAM needed by the I2C driver.

30.4.14 I2C Set-up

Table 421. 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	Handle to the I2C instance.
Description	Returns a handle to the I2C instance.

30.4.15 I2C Set Bit Rate

Table 422. 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 I2C instance. 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).

30.4.16 I2C Get Firmware Version

Table 423. 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.

30.4.17 I2C Get Status

Table 424. 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 I2C instance.
Return	Status code.
Description	Returns status code. The status code indicates the state of the I2C bus. Refer to I2C Status Code Table.

30.4.18 Error codes

Table 425. Error codes

Error Code	Description	Comment
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.
0x0006 0009	ERR_I2C_TIMEOUT	Reserved
0x0006 000A	ERR_I2C_BUFFER_UNDERFLOW	-

```
typedef enum
{
 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,
  /*0x00060009*/ ERR_I2C_TIMEOUT,
  /*0x0006000A*/ ERR_I2C_BUFFER_UNDERFLOW
} ErrorCode_t;
```

30.4.19 I2C Status code

Table 426. I2C Status code

Status code	Description
0	IDLE
1	MASTER_SEND
2	MASTER_RECEIVE
3	SLAVE_SEND
4	SLAVE_RECEIVE

30.4.20 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.

30.4.20.1 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. The error code is updated in the callback and the parameter n indicates the number of bytes transferred.

30.4.21 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_PARAM {
        uint32_t      num_bytes_send; /*!< No. of bytes to send */
        uint32_t      num_bytes_rec; /*!< No. of bytes to receive */
        uint8_t       *buffer_ptr_send;/*!< Pointer to send buffer */
        uint8_t       *buffer_ptr_rec;/*!< Pointer to receive buffer */
        I2C_CALLBK_T    func_pt; /*!< Callback function */
        uint8_t       stop_flag; /*!< Stop flag */
        uint8_t       dummy[3];
} I2C_PARAM_T;
```

The RESULT structure is as follows:

```
typedef struct I2C_RESULT {
        uint32_t n_bytes_sent;/*!< No. of bytes sent */
        uint32_t n_bytes_recd;/*!< No. of bytes received */
} I2C_RESULT_T;
```

30.4.22 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 CHIP_I2C_MODE {
        IDLE,                   /*!< IDLE state */
        MASTER_SEND,/*!< Master send state */
        MASTER_RECEIVE,/*!< Master Receive state */
        SLAVE_SEND, /*!< Slave send state */
        SLAVE_RECEIVE/*!< Slave receive state */
```

```
} CHIP_I2C_MODE_T;
```

30.5 Functional description

30.5.1 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 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.

30.5.2 I2C Master mode set-up

The I2C ROM Driver support polling and interrupts. In the master mode, 7-bit and 10-bit addressing are supported. The setup is as follows:

1. Allocate SRAM for the I2C ROM Driver by making a call to the i2c_get_mem_size() function.
2. Create the I2C handle by making a call to the i2c_setup() function.
3. Set the I2C operating frequency by making a call to the i2c_set_bitrate() function.

```
size_in_bytes = LPC_I2CD_API->i2c_get_mem_size();
i2c_handle = LPC_I2CD_API->i2c_setup(LPC_I2C_BASE, (uint32_t *)start_of_ram_block0 );
error_code = LPC_I2CD_API->i2c_set_bitrate((I2C_HANDLE_T*)i2c_handle, PCLK_in_Hz, bps_in_hz);
```

30.5.3 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.

31 25	24	23 17	16	15 9	8	7 1	0
Slave Address 3	GC	Slave Address 2	GC	Slave Address 1	GC	Slave Address 0	GC

Fig 93. I2C slave mode set-up address packing

```
size_in_bytes = LPC_I2CD_API->i2c_get_mem_size();
i2c_handle = LPC_I2CD_API->i2c_setup(LPC_I2C_BASE, (uint32_t *)start_of_ram_block0 );
error_code = LPC_I2CD_API->i2c_set_bitrate((I2C_HANDLE_T*)i2c_handle, PCLK_in_Hz, bps_in_hz);
error_code = LPC_I2CD_API->i2c_set_slave_addr((I2C_HANDLE_T*)i2c_handle, slave_addr, slave_addr_mask) ;
```

30.5.4 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 30.4.21.

- I2C_RESULT* contains the results after the function has executed.

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.

30.5.5 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 30.4.21.
- I2C_RESULT is a containing the results after the function executes. Section 30.4.21.

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.

UM10732

Chapter 31: LPC11U6x/E6x USART ROM API (USART0)

Rev. 1.8 — 19 December 2016 **User manual**

31.1 How to read this chapter

The USART ROM driver routines are available on all parts. The API described in this chapter provides routines to control the USART0 peripheral (see Chapter 11 “LPC11U6x/E6x USART0”).

See Chapter 32 “LPC11U6x/E6x USART ROM API (USART1/2/3/4)” for the API for USART1, USART2, USART3, and USART4.

31.2 Features

- Send and receive characters in asynchronous or synchronous mode
- Send and receive multiple characters (line) in asynchronous or synchronous UART mode
- Support for DMA mode

31.3 General description

The UART API handles sending and receiving characters using any of the USART block in asynchronous mode.

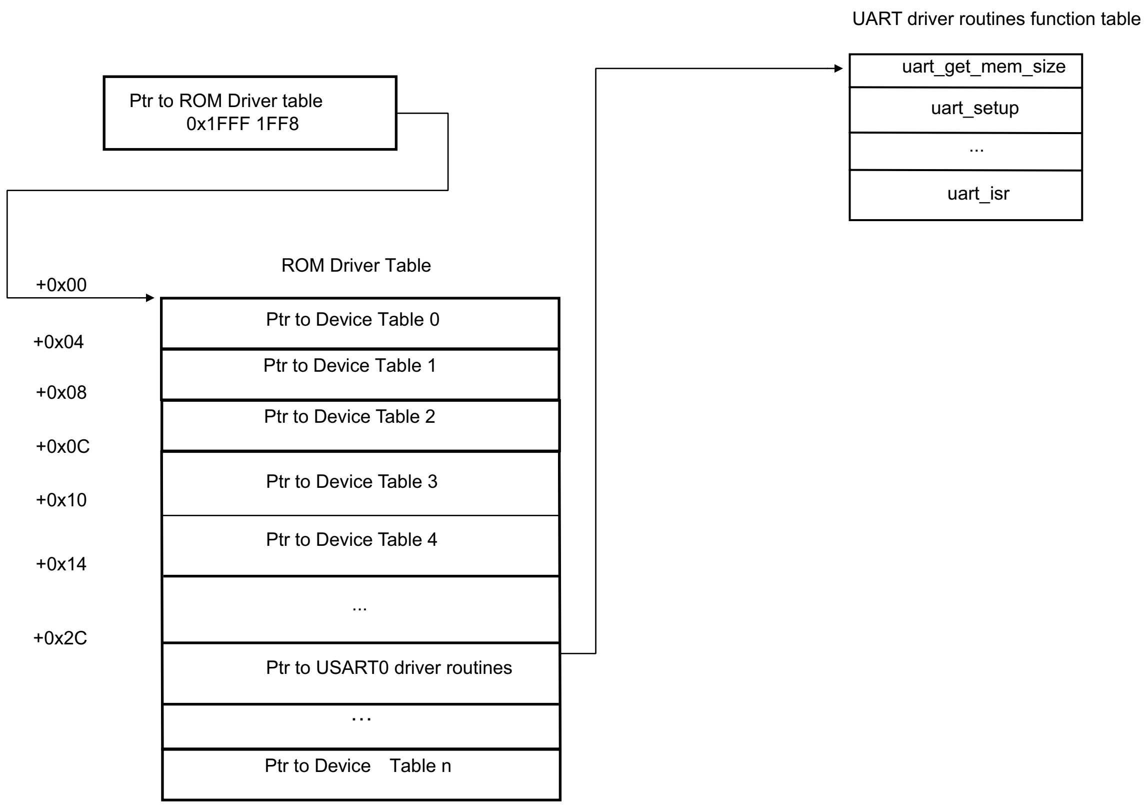


Fig 94. USART driver routines pointer structure (USART0)

31.4 API description

The UART API contains functions to send and receive characters via the USART0 block.

The API function table for the USART0 is identical with the table for USART1/2/3/4. However, the uart_init function returns the error code when used with the USART0 and the fractional rate generator value when used with USART1/2/3/4.

Table 427. UART API calls

API call	Description	Reference
uint32_t uart_get_mem_size(void) ;	UART get memory size for UART instance	Table 428
UART_HANDLE_T* uart_setup(uint32_t base_addr, uint8_t *ram) ;	UART set-up	Table 429
uint32_t uart_init(UART_HANDLE_T* handle, UART_CONFIG set);	UART init	Table 430
uint8_t uart_get_char(UART_HANDLE_T* handle);	UART get character	Table 431
void uart_put_char(UART_HANDLE_T* handle, uint8_t data);	UART put character	Table 432
uint32_t uart_get_line(UART_HANDLE_T* handle, UART_PARAM_T param);	UART get line	Table 433
uint32_t uart_put_line(UART_HANDLE_T* handle, UART_PARAM_T param);	UART put line	Table 434
void uart_isr(UART_HANDLE_T* handle);	UART interrupt service routine	Table 435

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);
           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);
          void (*uart_isr)(UART_HANDLE_T handle);
} UARTD_API_T  ;            // end of structure


#define LPC_UART0D_API((LPC_ROM_API)->pUART0D)
```

The ROM API table shown in Section 26.5.5 “ROM-based APIs” must be included in the code.

31.4.1 UART get memory size

Table 428. uart_get_mem_size

Routine	uart_get_mem_size
Prototype	uint32_t uart_get_mem_size(void) ;
Input parameter	None.
Return	Memory size in bytes.
Description	Get the memory size needed by one UART instance.

31.4.2 UART setup

Table 429. 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 is obtained from the uart_get_mem_size function.
Return	The handle to corresponding UART instance.
Description	Set up UART instance with provided memory and return the handle to this instance.

31.4.3 UART init

See Section 31.4.10.1 for the UART_CONFIG and Section 31.4.10.2 for UART_HANDLE_T variables. This function returns the error code.

Table 430. 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	Error code.
Description	Set up baud rate and operation mode, then enable UART.

31.4.4 UART get character

See Section 31.4.10.2 for UART_HANDLE_T variable. This function works in polling mode only.

Table 431. 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 character from UART. This functions is only returned after a character has been received.

31.4.5 UART put character

See Section 31.4.10.2 for UART_HANDLE_T variable. This function works in polling mode only.

Table 432. 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 character through UART. This function is only returned after a character has been sent.

31.4.6 UART get line

See Section 31.4.10.2 for UART_HANDLE_T variable and Section 31.4.10.3 for the PARAM_T variable.

If the length of the receive data is known, you can set the buffer to this length and the driver will adjust the receive FIFO trigger level automatically. If the data is received continuously (and terminated with <CR><LF> or <LF> for example) set the buffer size to one.

Table 433. 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.

31.4.7 UART put line

See Section 31.4.10.2 for UART_HANDLE_T variable and Section 31.4.10.3 for the PARAM_T variable.

The transmit FIFO trigger level is set to half of the buffer level by the UART driver.

Table 434. 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.

31.4.8 UART interrupt service routine

See Section 31.4.10.2 for UART_HANDLE_T variable.

Table 435. 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 UART interrupt must be enabled. This function is invoked by the user ISR.

31.4.9 Error codes

Table 436. Error codes

Return code	Error Code	Description
0x0008 0001	ERR_UART_RXD_BUSY	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
0x0008 0006	ERR_UART_BAUDRATE	Baudrate setting error

```
typedef enum
{
 ERR_UART_BASE = 0x00080000,
  /*0x00080001*/ ERR_UART_RXD_BUSY = ERR_UART_BASE+1,   //UART rxd is busy
  /*0x00080002*/ ERR_UART_TXD_BUSY,   //UART txd is busy
  /*0x00080003*/ ERR_UART_OVERRUN_FRAME_PARITY_NOISE, //overrun err, frame err, parity
            // err, RxNoise err
  /*0x00080004*/ ERR_UART_UNDERRUN,    //underrun err
  /*0x00080005*/ ERR_UART_PARAM,       //parameter error
  /*0x00080006*/ ERR_UART_BAUDRATE //baudrate setting error
} ErrorCode_t;
```

31.4.10 UART ROM driver variables

31.4.10.1 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
                        //bit4: 0 = SCLK cycles only when characters are being sent on TxD)
                        //        1 = SCLK runs continuously (characters can be received)
          uint16_t error_en;
                       //0 = Disable overrun error, parity error, framing error,
                        //    break indication or transmission error detection.
                        // 1 = Enable overrun error, parity error, framing error,
                        //    break indication or transmission error detection.
}
```

31.4.10.2 UART_HANDLE_T

The handle to the instance of the UART driver. Each UART has one handle, so there can be several handles for each UART block. 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
```

31.4.10.3 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, stop transfer when the buffer is full.
                                 // For uart_put_line function, stop transfer when the buffer is empty.
                                 // 0x01:  For uart_get_line function, stop transfer when
                                 // <CR><LF> characters have been received.
                                 // For uart_put_line function, transfer is stoppred 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.
                                 // NOTE: if (transfer_mode & 0x0F) != 0, transfer also stops
                                 // when all data in buffer has been transferred.
        uint8_t   driver_mode;
                                 // 0x00: Polling mode, function is blocked until transfer is
                                 // finished.
                                 // 0x01: Interrupt mode, function exit immediately. The callback
                                 //function
                                 // is invoked when transfer is finished.
                                 // 0x02: DMA mode (transfer_mode must be 0).
                                 // DMA req function is called for Uart DMA channel setup, then
                                 // DMA ISR indicate that transfer is finished.
        uint8_t dma_num;                 //DMA channel number in case DMA mode is enabled
        UART_CALLBK_T     callback_func_pt;
                                 // callback function
                                 // In case DMA mode is enabled, callback function is invoked
                                 // after transfer. If callback_func_pt = NULL, no DMA interrupt
                                 // is issued for this UART channel.
        uint32_t                 dma;    //DMA handler
}  UART_PARAM_T  ;
```

Remark: The DMA mode is enabled in UART_PARAM_T structure for both the uart_put_line and the uart_get_line simultaneously. To disable the DMA mode after a UART transfer, you must re-initialize the UART by calling the uart_init function first and then set up the UART_PARAM_T structure before the next transfer.

31.4.10.4 CALLBK_T

```
typedef void  (*CALLBK_T) (uint32_t   res0, uint32_t res1 ) ;
        //define callback func TYPE
        //res0: error code
```

```
//res1: number of bytes transferred
```

31.4.11 Functional description

31.4.11.1 Example (no DMA)

Send and receive characters in interrupt mode. Use the UART API as follows:

1. Assign the RXD and TXD functions to pins and set up the system clock, main clock, and UART clock dividers.
2. Global defines:

```
UART_HANDLE_T* uart_handle_0; //handle to UART API
UART_PARAM_T      param;

#define RAMBLOCK_H   10
uint32_t  start_of_ram_block[ RAMBLOCK_H ] ;

#define BUFFER_SIZE   100
uint32_t uart_buffer[BUFFER_SIZE];
```

3. Define configuration structure and initialize pointer to the UART API:

```
UART_CONFIG_T      uart_set;
#define LPC_UART0D_API  ((UARTD_API_T *) ((*(LPC_ROM_API_T * *)
         (ROM_DRIVER_BASE))->pUART0D))
```

4. Define some characters to send:

```
const uint8_t pattern4[] = "Test interrupt mode";
```

5. Initialize memory for one UART API and create handle:

```
 size_in_bytes = LPC_UART0D_API->uart_get_mem_size() ;
  if (     RAMBLOCK_H < (size_in_bytes /4 ) ) {
    return 1;
  }
uart_handle_0 = LPC_UART0D_API->uart_setup(LPC_UART0_BASE, (uint8_t*)start_of_ram_block);
```

6. Initialize UART, configure baud rate:

```
uart_set.sys_clk_in_hz = SystemCoreClock/4;
uart_set.baudrate_in_hz = BAUDRATE_IN_HZ;
uart_set.config = 1;// 8 bits data, no parity, 1 stop
uart_set.sync_mod = 0;
uart_set.error_en = 0;
```

7. Enable the UART interrupt in the NVIC.
8. Set up the UART parameter structure UART_PARAM_T:

```
param.driver_mode = 1; //INT mode
param.transfer_mode = 1; //stop transfer with \0,, <CR><LF> is sent out.
param.buffer = (uint8_t *)pattern4;
param.size = 100; //Max of buffer
param.callback_func_pt = put_callback;
```

9. Define the receive and transmit callback functions invoked when the transfer has finished:

   ```
   void  get_callback(uint32_t   err_code, uint32_t n ) {
     get_tag = 1;
     if (err_code != LPC_OK)
       while(1);
   }

   void  put_callback(uint32_t   err_code, uint32_t n ) {
     put_tag = 1;
     if (err_code != LPC_OK)
       while(1);
   }
   ```

10. Send some characters and stop transfer with \0. Then <CR><LF> is sent out.

    ```
    param.driver_mode = 1; //INT mode
    param.transfer_mode = 1; //stop transfer with \0,, <CR><LF> is sent out.
    param.buffer = (uint8_t *)pattern4;
    param.size = 100; //Max of buffer
    param.callback_func_pt = put_callback;
    put_tag = 0;
    LPC_UART0D_API->uart_put_line(uart_handle, &param);
    while(!put_tag);
    ```

11. Read five characters until buffer is full:

    ```
    param.transfer_mode = 0; //stop get when buffer is full
    param.buffer = (uint8_t *)buffer;
    param.size = 5; //size of buffer
    param.callback_func_pt = get_callback;
    get_tag = 0;
    LPC_UART0D_API->uart_get_line(uart_handle, &param);
    while(!get_tag);
    ```

UM10732

Chapter 32: LPC11U6x/E6x USART ROM API (USART1/2/3/4)

Rev. 1.8 — 19 December 2016 User manual

32.1 How to read this chapter

The USART ROM driver routines are available on all parts. The API described in this chapter provides routines to control the USART1, USART2, USART3, and USART4 peripherals (see Chapter 12 “LPC11U6x/E6x USART1/2/3/4”).

See Chapter 31 “LPC11U6x/E6x USART ROM API (USART0)” for the API for USART0.

32.2 Features

- Send and receive characters in asynchronous or synchronous mode
- Send and receive multiple characters (line) in asynchronous or synchronous UART mode
- Support for DMA mode

32.3 General description

The UART API handles sending and receiving characters using any of the USART blocks in asynchronous mode.

Remark: Because all USARTS (USART1, USART2, USART3, and USART4) share a common fractional divider configured in the SYSCON block, the uart_init routine returns the value for the common divider.

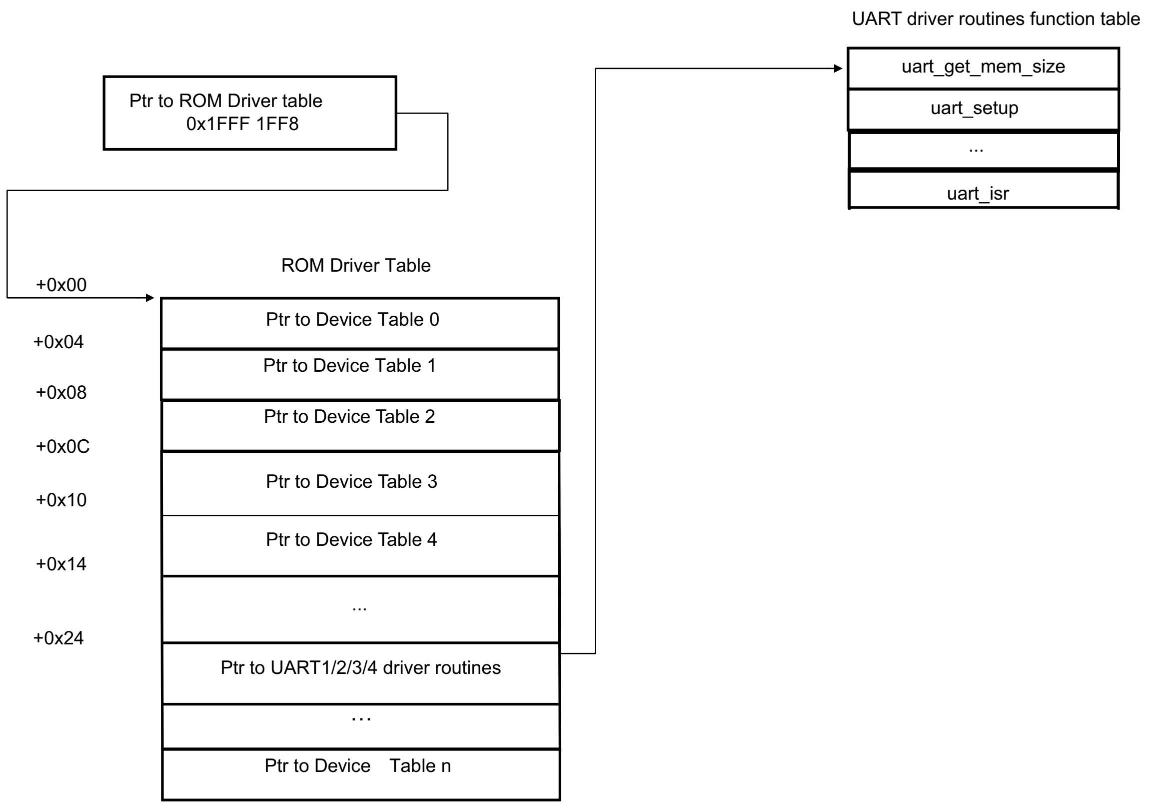


Fig 95. USART driver routines pointer structure (USART1/2/3/4)

The ROM API table shown in Section 26.5.5 “ROM-based APIs” must be included in the code.

32.4 API description

The UART API contains functions to send and receive characters via any of the USART blocks.

The API function table for the USART0 is identical with the table for USART1/2/3/4. However, the uart_init function returns the error code when used with the USART0 and the fractional rate generator value when used with USART1/2/3/4.

Table 437. UART API calls

API call	Description	Reference
uint32_t uart_get_mem_size(void) ;	UART get memory size for UART instance	Table 438
UART_HANDLE_T* uart_setup(uint32_t base_addr, uint8_t *ram) ;	UART set-up	Table 439
uint32_t uart_init(UART_HANDLE_T* handle, UART_CONFIG set);	UART init	Table 440
uint8_t uart_get_char(UART_HANDLE_T* handle);	UART get character	Table 441
void uart_put_char(UART_HANDLE_T* handle, uint8_t data);	UART put character	Table 442

Table 437. UART API calls

API call	Description	Reference
uint32_t uart_get_line(UART_HANDLE_T* handle, UART_PARAM_T param);	UART get line	Table 443
uint32_t uart_put_line(UART_HANDLE_T* handle, UART_PARAM_T param);	UART put line	Table 444
void uart_isr(UART_HANDLE_T* handle);	UART interrupt service routine	Table 445

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 only functions--//
            uint8_t (*uart_get_char)(UART_HANDLE_T handle);
            void (*uart_put_char)(UART_HANDLE_T handle, uint8_t data);
            //--polling or interrupt functions--//
            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);
            void (*uart_isr)(UART_HANDLE_T handle);
} UARTD_API_T  ;            // end of structure

#define LPC_UARTND_API((LPC_ROM_API)->pUARTND)
```

32.4.1 UART get memory size

Table 438. uart_get_mem_size

Routine	uart_get_mem_size
Prototype	uint32_t uart_get_mem_size(void) ;
Input parameter	None.
Return	Memory size in bytes.
Description	Get the memory size needed by one UART instance.

32.4.2 UART setup

Table 439. 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.

32.4.3 UART init

See Section 32.4.10.1 for the UART_CONFIG and Section 32.4.10.2 for UART_HANDLE_T variables. This function returns the value that must be written into the fractional baud rate generator register FRGCTRL in the SYSCON block to generate the desired baud rate.

Table 440. 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.

32.4.4 UART get character

See Section 32.4.10.2 for UART_HANDLE_T variable. This function works in polling mode only.

Table 441. 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.

32.4.5 UART put character

See Section 32.4.10.2 for UART_HANDLE_T variable. This function works in polling mode only.

Table 442. 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.

32.4.6 UART get line

See Section 32.4.10.2 for UART_HANDLE_T variable and Section 32.4.10.3 for the PARAM_T variable.

Table 443. 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.

32.4.7 UART put line

See Section 32.4.10.2 for UART_HANDLE_T variable and Section 32.4.10.3 for the PARAM_T variable.

Table 444. 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.

32.4.8 UART interrupt service routine

See Section 32.4.10.2 for UART_HANDLE_T variable.

Table 445. 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.

32.4.9 Error codes

Table 446. Error codes

Return code	**Error Code**	**Description**
0x0008 0001	ERR_UART_RXD_BUSY	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
0x0008 0006	ERR_UART_BAUDRATE	Baudrate setting error

```
typedef enum
```

```
{
 ERR_UART_BASE = 0x00080000,
  /*0x00080001*/ ERR_UART_RXD_BUSY = ERR_UART_BASE+1,   //UART rxd is busy
  /*0x00080002*/ ERR_UART_TXD_BUSY,   //UART txd is busy
  /*0x00080003*/ ERR_UART_OVERRUN_FRAME_PARITY_NOISE, //overrun err, frame err, parity
           // err, RxNoise err
  /*0x00080004*/ ERR_UART_UNDERRUN,    //underrun err
  /*0x00080005*/ ERR_UART_PARAM,       //parameter error
  /*0x00080006*/ ERR_UART_BAUDRATE //baudrate setting error
} ErrorCode_t;
```

32.4.10 UART ROM driver variables

32.4.10.1 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
}
```

32.4.10.2 UART_HANDLE_T

The handle to the instance of the UART driver. Each UART has one handle, so there can be several handles for each UART block. 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
```

32.4.10.3 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.
                                      // NOTE: if (transfer_mode & 0x0F) != 0, transfer also stops
                                      // when all data in buffer has been transferred.
uint8_t   driver_mode;
                                      // 0x00: Polling mode, function is blocked until transfer is
                                      // finished.
                                      // 0x01: Interrupt mode, function exit immediately, callback
                                      //function
                                      // is invoked when transfer is finished.
                                      // 0x02: DMA mode (transfer_mode must be 0).
                                      // DMA req function is called for Uart DMA channel setup, then
                                      // DMA ISR indicate that transfer is finished.
uint8_t dma_num;                      //DMA channel number in case DMA mode is enabled
UART_CALLBK_T     callback_func_pt;
                                      // callback function
                                      // In case DMA mode is enabled, callback function is invoked
                                      // after transfer. If callback_func_pt = NULL, no DMA interrupt
                                      // is issued for this UART channel.
uint32_t                    dma;     //DMA handler
}  UART_PARAM_T  ;
```

32.4.10.4 CALLBK_T

```
typedef void  (*CALLBK_T) (uint32_t   res0, uint32_t res1 ) ;
          //define callback func TYPE
          //res0: error code
          //res1: number of bytes transferred
```

32.4.11 Functional description

32.4.11.1 Example (no DMA)

Send and receive characters in interrupt mode. Use the UART API as follows:

1. Assign the RXD and TXD functions to pins and set up the system clock, main clock, and UART clock dividers.

2. Global defines:

```
UART_HANDLE_T* uart_handle_0; //handle to UART API

UART_PARAM_T      param;

#define RAMBLOCK_H   10
uint32_t  start_of_ram_block[ RAMBLOCK_H ] ;

#define BUFFER_SIZE   100
uint32_t uart_buffer[BUFFER_SIZE];
```

3. Define configuration structure and initialize pointer to the UART API:

```
UART_CONFIG_T      uart_set;
#define LPC_UARTND_API  ((UARTD_API_T *) ((*(LPC_ROM_API_T * *)
          (ROM_DRIVER_BASE))->pUARTND))
```

4. Define some characters to send:

```
const uint8_t pattern4[] = "Test interrupt mode";
```

5. Initialize memory for one UART API and create handle:

```
 size_in_bytes = LPC_UARTND_API->uart_get_mem_size() ;
  if (      RAMBLOCK_H < (size_in_bytes /4 ) ) {
    return 1;
  }
uart_handle_0 = LPC_UARTND_API->uart_setup(LPC_UART0_BASE, (uint8_t
           *)start_of_ram_block);
```

6. Initialize UART, configure baud rate. Use the result of the UART initialization to configure the fractional baud rate generator register FRGCTRL in the SYSCON block (the lower 8 bits of this register must always be set to 0xFF):

```
uart_set.sys_clk_in_hz = SystemCoreClock/4;
uart_set.baudrate_in_hz = BAUDRATE_IN_HZ;
uart_set.config = 1;// 8 bits data, no parity, 1 stop
uart_set.sync_mod = 0;
uart_set.error_en = 0;
LPC_SYSCON->UARTFRGMULT = (LPC_UARTND_API->uart_init(uart_handle_0,
          &uart_set)<<0x08)|0xFF;
```

7. Enable the UART interrupt in the NVIC.
8. Set up the UART parameter structure UART_PARAM_T:

```
param.driver_mode = 1; //INT mode
param.transfer_mode = 1; //stop transfer with \0,, <CR><LF> is sent out.
param.buffer = (uint8_t *)pattern4;
param.size = 100; //Max of buffer
param.callback_func_pt = put_callback;
```

9. Define the receive and transmit callback functions invoked when the transfer has finished:

```
void  get_callback(uint32_t   err_code, uint32_t n ) {
  get_tag = 1;
```

```
  if (err_code != LPC_OK)
    while(1);
}

void  put_callback(uint32_t   err_code, uint32_t n ) {
  put_tag = 1;
  if (err_code != LPC_OK)
    while(1);
}
```

10. Send some characters and stop transfer with \0. Then <CR><LF> is sent out.

```
param.driver_mode = 1; //INT mode
param.transfer_mode = 1; //stop transfer with \0,, <CR><LF> is sent out.
param.buffer = (uint8_t *)pattern4;
param.size = 100; //Max of buffer
param.callback_func_pt = put_callback;
put_tag = 0;
LPC_UARTND_API->uart_put_line(uart_handle, &param);
while(!put_tag);
```

11. Read five characters until buffer is full:

```
param.transfer_mode = 0; //stop get when buffer is full
param.buffer = (uint8_t *)buffer;
param.size = 5; //size of buffer
param.callback_func_pt = get_callback;
get_tag = 0;
LPC_UARTND_API->uart_get_line(uart_handle, &param);
while(!get_tag);
```

UM10732

Chapter 33: LPC11U6x/E6x DMA ROM API

Rev. 1.8 — 19 December 2016 **User manual**

33.1 How to read this chapter

The DMA ROM driver routines are available on all parts.

33.2 Features

- DMA set-up
- DMA channel control
- DMA transfers

33.3 General description

The DMA API handles DMA set-up and transfers.

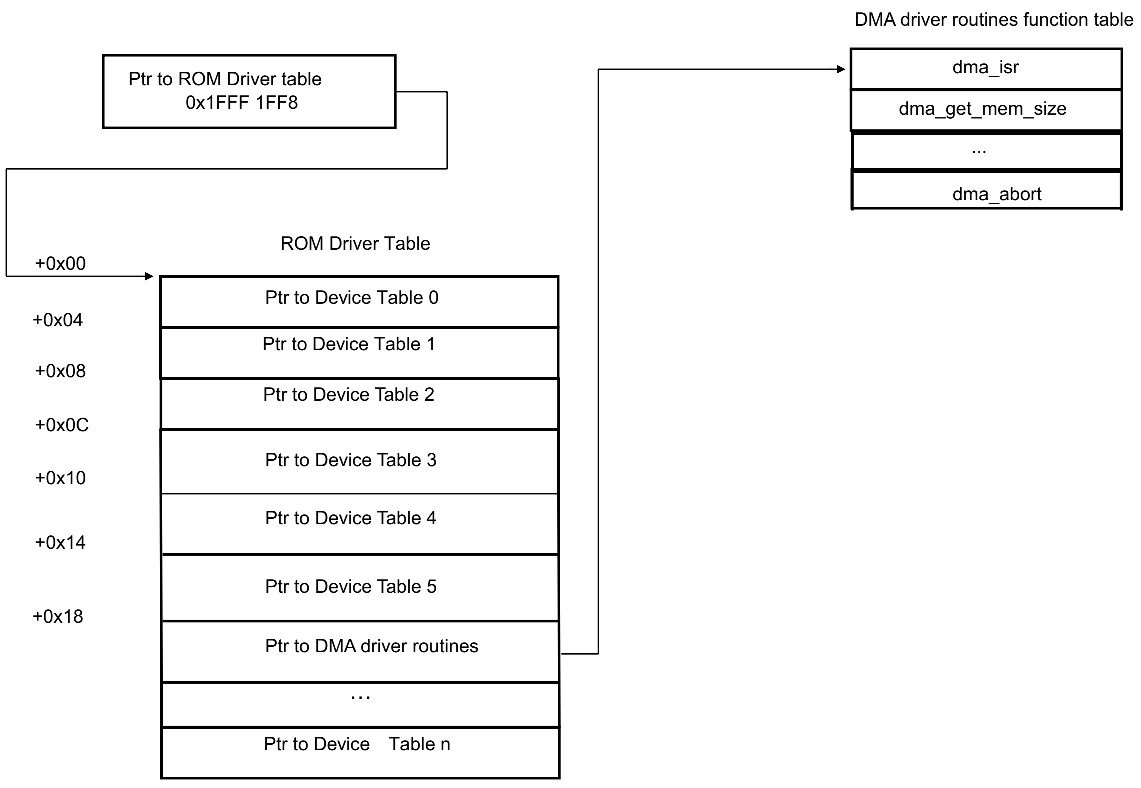


Fig 96. DMA driver routines pointer structure

The ROM API table shown in Section 26.5.5 “ROM-based APIs” must be included in the code.

33.4 API description

The DMA API contains functions set up and operate the DMA controller.

Table 447. DMA API calls

API call	Description	Reference
Interrupt service		
`void  dma_isr(DMA_HANDLE_T* handle);`	Interrupt service routine	Table 456
Initialization		
`uint32_t  dma_get_mem_size(void) ;`	Get memory size needed for DMA.	Table 448
`DMA_HANDLE_T* dma_setup(uint32_t  base_addr,  uint8_t   *ram);`	Set up DMA.	Table 449
DMA channel operations		
`uint32_t dma_init(DMA_HANDLE_T* handle, DMA_CHANNEL_T *channel, DMA _TASK_T *task);`	Enable DMA channel and set up basic DMA transfer.	Table 450
`uint32_t dma_task_link(DMA_HANDLE_T* handle, DMA _TASK_T *task, uint8_t valid);`	Create linked transfer.	Table 451
`void dma_set_valid(DMA_HANDLE_T* handle, unit8_t chl_num);`	Set a task to valid.	Table 452
`void  dma_pause(DMA_HANDLE_T* handle, unit8_t chl_num);`	Pause DMA transfer on one channel.	Table 453
`void  dma_unpause(DMA_HANDLE_T* handle, unit8_t chl_num);`	Resume DMA transfer.	Table 454
`void  dma_abort(DMA_HANDLE_T* handle, unit8_t chl_num);`	Cancel DMA transfer on one channel.	Table 455

The following structure must be defined to use the DMA API:

```
typedef struct  DMAD_API {    // index of all the DMA driver functions
          void                (*dma_isr)(DMA_HANDLE_T*  handle);
          uint32_t          (*dma_get_mem_size)( void);
          DMA_HANDLE_T* (*dma_setup)( uint32_t base_addr,  uint8_t *ram );
          ErrorCode_t   (*dma_init)( DMA_HANDLE_T*  handle, DMA_CHANNEL_T *channel, DMA_TASK_T
          *task);
          ErrorCode_t   (*dma_link)( DMA_HANDLE_T* handle, DMA_TASK_T *task, uint8_t valid);
          ErrorCode_t   (*dma_set_valid)( DMA_HANDLE_T* handle, uint8_t chl_num);
          ErrorCode_t   (*dma_pause)( DMA_HANDLE_T* handle, uint8_t chl_num);
          ErrorCode_t   (*dma_unpause)( DMA_HANDLE_T* handle, uint8_t chl_num);
          ErrorCode_t   (*dma_abort)( DMA_HANDLE_T* handle, uint8_t chl_num);
} DMAD_API_T  ;

#define LPC_DMAD_API ((LPC_ROM_API)->pDMAD)
```

33.4.1 DMA get memory size

Table 448. dma_get_mem_size

Routine	dma_get_mem_size
Prototype	`uint32_t  dma_get_mem_size(void);`

Table 448. dma_get_mem_size

Routine	dma_get_mem_size
Input parameter	None.
Return	Memory size in bytes.
Description	The memory size for the DMA instance.

33.4.2 DMA set-up

Table 449. dma_setup

Routine	dma_setup
Prototype	`DMA_HANDLE_T* dma_setup(uint32_t base_addr, uint8_t *ram) ;`
Input parameter	base_addr: Base address of register for DMA block. ram: Pointer to the memory space for the DMA Channel descriptor map used by the DMA instance. The size is obtained by the dma_get_mem_size() function.
Return	0: the alignment of address for DMA descriptor map is not correct. Others: The handle to corresponding DMA instance.
Description	Sets up DMA instance with provided memory. Checks the alignment of address for Channel descriptor map according to the number of channels and returns the handle of this instance if the address alignment is correct.

33.4.3 DMA init

After the handler is initialized, the DMA channel API is invoked to set up a channel for data transfer.

Table 450. dma_init

Routine	dma_init
Prototype	`uint32_t dma_init(DMA_HANDLE_T* handle, DMA_CHANNEL_T *channel, DMA _TASK_T *task);`
Input parameter	handle: The handler to the DMA instance. channel: The pointer to the structure for DMA channel setup. task: The pointer to the structure for basic transfer task setup.
Return	Error code.
Description	Enables the DMA channel and sets up a basic transfer task. If no further DMA channel operation API is invoked, a single buffer DMA transfer is performed with DMA request or trigger.

33.4.4 DMA link

Table 451. dma_link

Routine	dma_link
Prototype	uint32_t dma_task_link(DMA_HANDLE_T* handle, DMA _TASK_T *task, uint8_t valid);
Input parameter	handle: The handler to the DMA instance. task: The pointer to the structure for transfer task setup. Valid: valid status of task 0: The task is not enabled for DMA, calling dma_task_valid is needed to process this task. 1: The task is valid for DMA, DMA will process this task in case request or trigger is fulfilled.
Return	Error code.
Description	Link an additional transfer task to the DMA channel enabled previously by calling dma_setup.

33.4.5 DMA set valid transfer

Table 452. dma_set_valid

Routine	dma_set_valid
Prototype	void dma_set_valid(DMA_HANDLE_T* handle, unit8_t chl_num);
Input parameter	handle: The handler to the DMA instance. chl_num: DMA channel number to be enabled.
Return	None
Description	If the DMA fetches an invalid transfer task, DMA will not process this task until this function is called. If the DMA is transferring date for a valid task, calling this function will make next invalid task served immediately when it is fetched by the DMA.

33.4.6 DMA pause transfer

Table 453. dma_pause

Routine	dma_pause
Prototype	void dma_pause(DMA_HANDLE_T* handle, unit8_t chl_num);
Input parameter	handle: The handler to the DMA instance. chl_num: DMA channel number to be paused.
Return	None
Description	Pauses one DMA channel transfer.

33.4.7 DMA resume transfer

Table 454. dma_unpause

Routine	dma_unpause
Prototype	void dma_unpause(DMA_HANDLE_T* handle, unit8_t chl_num);

Table 454. dma_unpause

Routine	dma_unpause
Input parameter	handle: The handler to the DMA instance. chl_num: DMA channel number to be resumed.
Return	None
Description	Resume one DMA channel transfer that has been paused previously.

33.4.8 DMA abort transfer

Table 455. dma_abort

Routine	dma_abort
Prototype	`void dma_abort(DMA_HANDLE_T* handle, unit8_t chl_num);`
Input parameter	handle: The handler to the DMA instance. chl_num: DMA channel number to be aborted.
Return	None
Description	Cancel one DMA channel transfer tasks. Recovering is impossible.

33.4.9 DMA interrupt service routine

Table 456. dma_isr

Routine	dma_isr
Prototype	`void dma_isr(DMA_HANDLE_T* handle);`
Input parameter	handle: The handler to the DMA instance.
Return	None.
Description	DMA interrupt service routine.

33.4.10 Error codes

Table 457. Error codes

Return code	Error Code	Description
0x000D 0001	ERR_DMA_ERROR_INT	-
0x000D 0002	ERR_DMA_CHANNEL_NUMBER	-
0x000D 0003	ERR_DMA_CHANNEL_DISABLED	-
0x000D 0004	ERR_DMA_BUSY	-
0x000D 0005	ERR_DMA_NOT_ALIGNMENT	-
0x000D 0006	ERR_DMA_PING_PONG_EN	Reload bit already set causing ping-pong mode error
0x000D 0007	ERR_DMA_CHANNEL_VALID_PENDING	-

```
ERR_DMA_BASE = 0x000D0000,
 /*0x000D0001*/ ERR_DMA_ERROR_INT=ERR_DMA_BASE+1,
 /*0x000D0002*/ ERR_DMA_CHANNEL_NUMBER,
 /*0x000D0003*/ ERR_DMA_CHANNEL_DISABLED,
 /*0x000D0004*/ ERR_DMA_BUSY,
 /*0x000D0005*/ ERR_DMA_NOT_ALIGNMENT,
 /*0x000D0006*/ ERR_DMA_PING_PONG_EN,
```

```
    /*0x000D0007*/ ERR_DMA_CHANNEL_VALID_PENDING
```

33.4.11 DMA ROM driver variables

33.4.11.1 DMA_CHANNEL_T channel configuration structure

```
typedef struct  DMA_CHANNEL {
        uint8_t event;       // event type selection for DMA transfer
                //0: software request
                //1: peripheral request
                //2: hardware trigger
                //others: reserved
        uint8_t hd_trigger; //In case hardware trigger is enabled, the trigger burst is  // set up here.
        //Rising edge triggered is fixed.
        //bit0~bit3: burst size
                //0: burst size =1, 1: 21, 2: 22... 10: 1024, others: reserved.
        //bit4: Source Burst Wrap
                //0: Source burst wrapping is not enabled
                //1: Source burst wrapping is enabled
        //bit5: Destination Burst Wrap
                //0: Destination burst wrapping is not enabled
                //1: Destination burst wrapping is enabled
        //bit6: Trigger Burst
        //0: Hardware trigger cause a single transfer
        //1: Hardware trigger cause a burst transfer
        //bit7: reserved
        uint8_t Priority;      //priority level
        //0 -> 7: Highest priority ->  Lowest priority.
        //other: reserved.
CALLBK_T     callback_func_pt; // callback function, Callback function is only
        // invoked when INTA or INTB is enabled.
}  DMA_CHANNEL_T  ;
```

33.4.11.2 DMA_HANDLE_T

The handler to the instance of DMA driver. This handle is created by Init API and used by the other function in the DMA driver.

```
typedef  void     DMA_HANDLE_T   ;    // define TYPE for DMA handle pointer
```

33.4.11.3 DMA_TASK_T

```
typedef struct  DMA_ TASK {
        unit8_t ch_num  // DMA channel number.
        uint8_t config;     //configuration of this task
                //bit0: Ping_Pong transfer
                        //0: Not Ping_Pong transfer
                        //1: Linked with previous task for Ping_Pong transfer
                //bit1: Software Trigger
                        //0: the trigger for this channel is not set.
                        //1: the trigger for this channel is set immediately.
                //bit2:  Clear Trigger
                        //0: The trigger is not cleared when this task is finished.
```

```
                    //1: The trigger is cleared when this task is finished.
          //bit3:  Select INTA
                    //0: No IntA.
                    //1: The IntB flag for this channel will be set when this task is
                    // finished.
          //bit4:  Select INTB
                    //0: No IntB.
                    //1: The IntB flag for this channel will be set when this task is finished.
          //bit5~bit7: reserved
     uint8_t   data_type;
          //bit0~bit1: Data width. 0: 8-bit, 1: 16-bit, 2: 32-bit, 3: reserved
          //bit2~bit3: How is source address incremented?
                    //0: The source address is not incremented for each transfer.
                    //1: The source address is incremented by the amount specified by
                    // Width for     each transfer.
                    //2: The source address is incremented by 2 times the amount specified
                    // by Width for each transfer.
                    //3: The source address is incremented by 4 times the amount specified
                    // by Width for each transfer.
          //bit4~bit5: How is the destination address incremented?
                    //0: The destination address is not incremented for each transfer.
                    //1: The destination address is incremented by the amount specified by
                    // Width for each transfer.
                    //2: The destination address is incremented by 2 times the amount
                    // specified by Width for each transfer.
                    //3: The destination address is incremented by 4 times the amount
                    // specified by Width for each transfer.
          //bit6~bit7: reserved.
     uint16_t  data_length; //0: 1 transfer, 1: 2 transfer, ... 1023: 1024 transfer.
          //Others: reserved.
     uint32_t  src;      // Source data end address
     uint32_t  dst;      // Destination end address
     uint32_t task_addr;    //the address of RAM for saving this task.
          //(NOTE: each task need 16 bytes RAM for storing configuration,
          //  and DMA API could set it according user input parameter,
          //  but it is responsible of user to allocate this RAM space and
          //  make sure that the base address must be 16-byte alignment.
          //  And if user has setup the next_tast(!=0), the dma_task_link
          //  must be called for this task setup, otherwise unpredictable error will
          // happen.)
} DMA_TASK_T ;
```

33.4.11.4 CALLBK_T

```
typedef void  (*CALLBK_T) (uint32_t   res0, uint32_t res1 ) ;
     //define callback func TYPE
     //res0: error code
     //res1: : 0 = INTA is issued, 1 = INTB is issued
```

UM10732

Chapter 34: LPC11U6x USB ROM API

Rev. 1.8 — 19 December 2016 **User manual**

34.1 How to read this chapter

The USB ROM driver routines are available on all parts.

34.2 Features

- ROM-base USB drivers
- Communication Device Class (CDC) device class
- Human Interface Device (HID) device class
- Mass storage device class

34.3 General description

The boot ROM contains a USB driver to simplify the USB application development. The USB driver implements the Communication Device Class (CDC), the Human Interface Device (HID), and the Mass Storage Device (MSC) device class. The USB on-chip drivers support composite device.

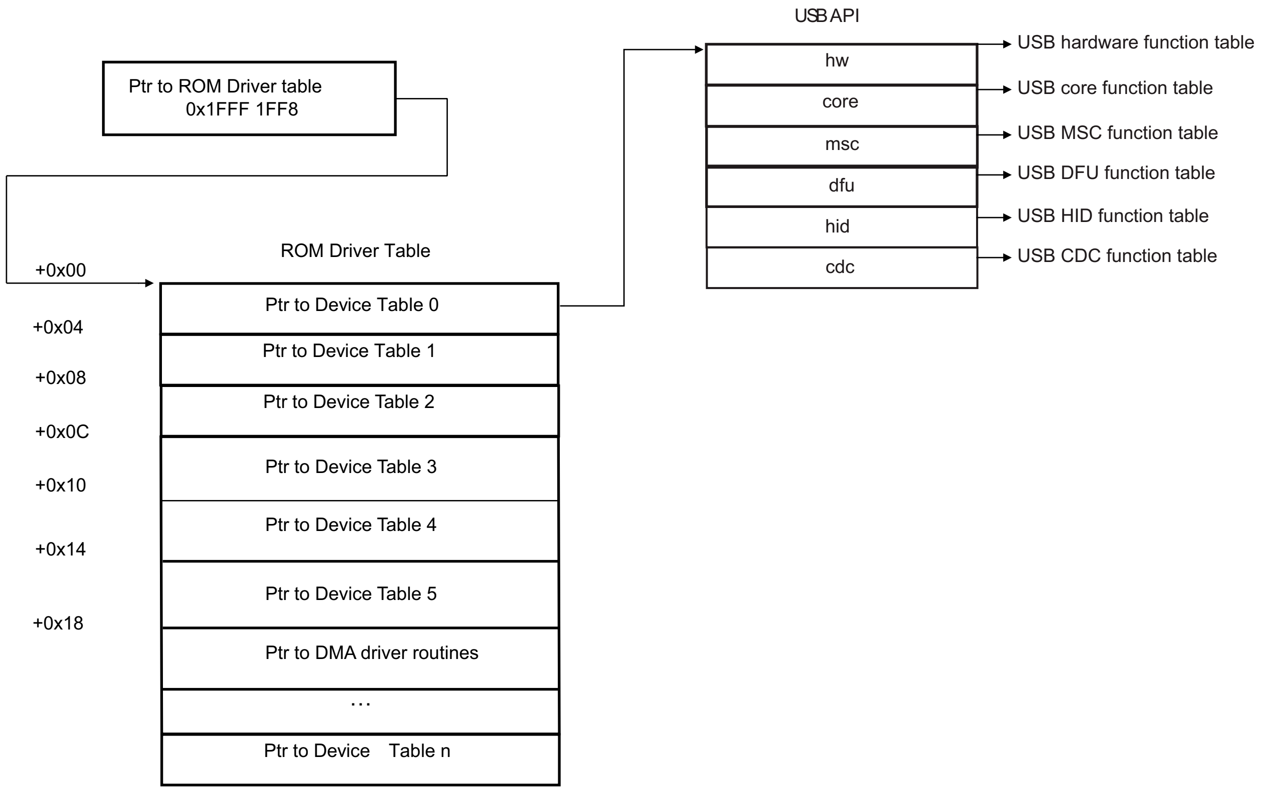


Fig 97. USB driver routines pointer structure

34.3.1 USB driver functions

The USB device driver ROM API consists of the following modules:

- Communication Device Class (CDC) function driver. This module contains an internal implementation of the USB CDC Class. User applications can use this class driver instead of implementing the CDC-ACM class manually via the low-level USBD_HW and USBD_Core APIs. This module is designed to simplify the user code by exposing only the required interface needed to interface with Devices using the USB CDC-ACM Class.
    - Communication Device Class function driver initialization parameter data structure (Table 485 “USBD_CDC_INIT_PARAM class structure”).
    - CDC class API functions structure. This module exposes functions which interact directly with USB device controller hardware (Table 484 “USBD_CDC_API class structure”).
- USB core layer
    - struct (Table 481 “_WB_T class structure”)
    - union (Table 458 “__WORD_BYTE class structure”)
    - struct (Table 459 “_BM_T class structure”)
    - struct (Table 472 “_REQUEST_TYPE class structure”)

  - struct (Table 479 “_USB_SETUP_PACKET class structure”)
  - struct (Table 475 “_USB_DEVICE_QUALIFIER_DESCRIPTOR class structure”)
  - struct USB device descriptor
  - struct (Table 475 “_USB_DEVICE_QUALIFIER_DESCRIPTOR class structure”)
  - struct USB configuration descriptor
  - struct (Table 477 “_USB_INTERFACE_DESCRIPTOR class structure”)
  - struct USB endpoint descriptor
  - struct (Table 480 “_USB_STRING_DESCRIPTOR class structure”)
  - struct (Table 473 “_USB_COMMON_DESCRIPTOR class structure”)
  - struct (Table 478 “_USB_OTHER_SPEED_CONFIGURATION class structure”)
  - USB descriptors data structure (Table 474 “_USB_CORE_DESCS_T class structure”)
  - USB device stack initialization parameter data structure (Table 483 “USBD_API_INIT_PARAM class structure”).
  - USB device stack core API functions structure (Table 486 “USBD_CORE_API class structure”).
- Device Firmware Upgrade (DFU) class function driver
  - DFU descriptors data structure (Table 488 “USBD_DFU_INIT_PARAM class structure”).
  - DFU class API functions structure. This module exposes functions which interact directly with the USB device controller hardware (Table 487 “USBD_DFU_API class structure”).
- HID class function driver
  - struct (Table 467 “_HID_DESCRIPTOR class structure”).
  - struct (Table 469 “_HID_REPORT_T class structure”).
  - USB descriptors data structure (Table 490 “USBD_HID_INIT_PARAM class structure”).
  - HID class API functions structure. This structure contains pointers to all the functions exposed by the HID function driver module (Table 491 “USBD_HW_API class structure”).
- USB device controller driver
  - Hardware API functions structure. This module exposes functions which interact directly with the USB device controller hardware (Table 491 “USBD_HW_API class structure”).
- Mass Storage Class (MSC) function driver
  - Mass Storage Class function driver initialization parameter data structure (Table 493).
  - MSC class API functions structure. This module exposes functions which interact directly with the USB device controller hardware (Table 492).

34.3.2 Calling the USB device driver

A fixed location in ROM contains a pointer to the ROM driver table i.e. 0x1FFF 1FF8. The ROM driver table contains a pointer to the USB driver table. Pointers to the various USB driver functions are stored in this table. USB driver functions can be called by using a C structure. Figure 97 illustrates the pointer mechanism used to access the on-chip USB driver.

```
typedef struct USBD_API
{
        const USBD_HW_API_T* hw;
        const USBD_CORE_API_T* core;
        const USBD_MSC_API_T* msc;
        const USBD_DFU_API_T* dfu;
        const USBD_HID_API_T* hid;
        const USBD_CDC_API_T* cdc;
        const uint32_t* reserved6;
        const uint32_t version;
} USBD_API_T;
```

The ROM API table shown in Section 26.5.5 “ROM-based APIs” must be included in the code.

34.4 USB API

34.4.1 __WORD_BYTE

Table 458. __WORD_BYTE class structure

Member	Description
W	`uint16_t __WORD_BYTE::W` data member to do 16 bit access
WB	`WB_TWB_T __WORD_BYTE::WB` data member to do 8 bit access

34.4.2 _BM_T

Table 459. _BM_T class structure

Member	Description
Recipient	`uint8_t _BM_T::Recipient` Recipient type.
Type	`uint8_t _BM_T::Type` Request type.
Dir	`uint8_t _BM_T::Dir` Direction type.

34.4.3 _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR

Table 460. _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR class structure

Member	Description
bFunctionLength	uint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bFunctionLength
bDescriptorType	uint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bDescriptorType
bDescriptorSubtype	uint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bDescriptorSubtype
bmCapabilities	uint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bmCapabilities

34.4.4 _CDC_CALL_MANAGEMENT_DESCRIPTOR

Table 461. _CDC_CALL_MANAGEMENT_DESCRIPTOR class structure

Member	Description
bFunctionLength	uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bFunctionLength
bDescriptorType	uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bDescriptorType
bDescriptorSubtype	uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bDescriptorSubtype
bmCapabilities	uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bmCapabilities
bDataInterface	uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bDataInterface

34.4.5 _CDC_HEADER_DESCRIPTOR

Table 462. _CDC_HEADER_DESCRIPTOR class structure

Member	Description
bFunctionLength	uint8_t _CDC_HEADER_DESCRIPTOR::bFunctionLength
bDescriptorType	uint8_t _CDC_HEADER_DESCRIPTOR::bDescriptorType
bDescriptorSubtype	uint8_t _CDC_HEADER_DESCRIPTOR::bDescriptorSubtype
bcdCDC	uint16_t _CDC_HEADER_DESCRIPTOR::bcdCDC

34.4.6 _CDC_LINE_CODING

Table 463. _CDC_LINE_CODING class structure

Member	Description
dwDTERate	uint32_t _CDC_LINE_CODING::dwDTERate
bCharFormat	uint8_t _CDC_LINE_CODING::bCharFormat
bParityType	uint8_t _CDC_LINE_CODING::bParityType
bDataBits	uint8_t _CDC_LINE_CODING::bDataBits

34.4.7 _CDC_UNION_1SLAVE_DESCRIPTOR

Table 464. _CDC_UNION_1SLAVE_DESCRIPTOR class structure

Member	Description
sUnion	CDC_UNION_DESCRIPTORCDC_UNION_DESCRIPTOR _CDC_UNION_1SLAVE_DESCRIPTOR::sUnion
bSlaveInterfaces	uint8_t _CDC_UNION_1SLAVE_DESCRIPTOR::bSlaveInterfaces[1][1]

34.4.8 _CDC_UNION_DESCRIPTOR

Table 465. _CDC_UNION_DESCRIPTOR class structure

Member	Description
bFunctionLength	uint8_t _CDC_UNION_DESCRIPTOR::bFunctionLength
bDescriptorType	uint8_t _CDC_UNION_DESCRIPTOR::bDescriptorType
bDescriptorSubtype	uint8_t _CDC_UNION_DESCRIPTOR::bDescriptorSubtype
bMasterInterface	uint8_t _CDC_UNION_DESCRIPTOR::bMasterInterface

34.4.9 _DFU_STATUS

Table 466. _DFU_STATUS class structure

Member	Description
bStatus	uint8_t _DFU_STATUS::bStatus
bwPollTimeout	uint8_t _DFU_STATUS::bwPollTimeout[3][3]
bState	uint8_t _DFU_STATUS::bState
iString	uint8_t _DFU_STATUS::iString

34.4.10 _HID_DESCRIPTOR

HID class-specific HID Descriptor.

Table 467. _HID_DESCRIPTOR class structure

Member	Description
bLength	uint8_t _HID_DESCRIPTOR::bLength Size of the descriptor, in bytes.
bDescriptorType	uint8_t _HID_DESCRIPTOR::bDescriptorType Type of HID descriptor.
bcdHID	uint16_t _HID_DESCRIPTOR::bcdHID BCD encoded version that the HID descriptor and device complies to.
bCountryCode	uint8_t _HID_DESCRIPTOR::bCountryCode Country code of the localized device, or zero if universal.
bNumDescriptors	uint8_t _HID_DESCRIPTOR::bNumDescriptors Total number of HID report descriptors for the interface.
DescriptorList	PRE_PACK struct POST_PACK _HID_DESCRIPTOR::_HID_DESCRIPTOR_LISTPRE_PACK struct POST_PACK _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST _HID_DESCRIPTOR::DescriptorList[1][1] Array of one or more descriptors

34.4.11 _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST

Table 468. _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST class structure

Member	Description
bDescriptorType	uint8_t _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST::bDescriptorType Type of HID report.
wDescriptorLength	uint16_t _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST::wDescriptorLength Length of the associated HID report descriptor, in bytes.

34.4.12 _HID_REPORT_T

HID report descriptor data structure.

Table 469. _HID_REPORT_T class structure

Member	Description
len	uint16_t _HID_REPORT_T::len Size of the report descriptor in bytes.
idle_time	uint8_t _HID_REPORT_T::idle_time This value is used by stack to respond to Set_Idle & GET_Idle requests for the specified report ID. The value of this field specified the rate at which duplicate reports are generated for the specified Report ID. For example, a device with two input reports could specify an idle rate of 20 milliseconds for report ID 1 and 500 milliseconds for report ID 2.
__pad	uint8_t _HID_REPORT_T::__pad Padding space.
desc	uint8_t * _HID_REPORT_T::desc Report descriptor.

34.4.13 _MSC_CBW

Table 470. _MSC_CBW class structure

Member	Description
dSignature	uint32_t _MSC_CBW::dSignature
dTag	uint32_t _MSC_CBW::dTag
dDataLength	uint32_t _MSC_CBW::dDataLength
bmFlags	uint8_t _MSC_CBW::bmFlags
bLUN	uint8_t _MSC_CBW::bLUN
bCBLength	uint8_t _MSC_CBW::bCBLength
CB	uint8_t _MSC_CBW::CB[16][16]

34.4.14 _MSC_CSW

Table 471. _MSC_CSW class structure

Member	Description
dSignature	uint32_t _MSC_CSW::dSignature
dTag	uint32_t _MSC_CSW::dTag
dDataResidue	uint32_t _MSC_CSW::dDataResidue
bStatus	uint8_t _MSC_CSW::bStatus

34.4.15 _REQUEST_TYPE

Table 472. _REQUEST_TYPE class structure

Member	Description
B	uint8_t _REQUEST_TYPE::B byte wide access member
BM	BM_TBM_T _REQUEST_TYPE::BM bitfield structure access member

34.4.16 _USB_COMMON_DESCRIPTOR

Table 473. _USB_COMMON_DESCRIPTOR class structure

Member	Description
bLength	uint8_t _USB_COMMON_DESCRIPTOR::bLength Size of this descriptor in bytes
bDescriptorType	uint8_t _USB_COMMON_DESCRIPTOR::bDescriptorType Descriptor Type

34.4.17 _USB_CORE_DESCS_T

USB descriptors data structure.

Table 474. _USB_CORE_DESCS_T class structure

Member	Description
device_desc	uint8_t * _USB_CORE_DESCS_T::device_desc Pointer to USB device descriptor
string_desc	uint8_t * _USB_CORE_DESCS_T::string_desc Pointer to array of USB string descriptors
full_speed_desc	uint8_t * _USB_CORE_DESCS_T::full_speed_desc Pointer to USB device configuration descriptor when device is operating in full speed mode.
high_speed_desc	uint8_t * _USB_CORE_DESCS_T::high_speed_desc Pointer to USB device configuration descriptor when device is operating in high speed mode. For full-speed only implementation this pointer should be same as full_speed_desc.
device_qualifier	uint8_t * _USB_CORE_DESCS_T::device_qualifier Pointer to USB device qualifier descriptor. For full-speed only implementation this pointer should be set to null (0).

34.4.18 _USB_DEVICE_QUALIFIER_DESCRIPTOR

Table 475. _USB_DEVICE_QUALIFIER_DESCRIPTOR class structure

Member	Description
bLength	uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bLength Size of descriptor
bDescriptorType	uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDescriptorType Device Qualifier Type
bcdUSB	uint16_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bcdUSB USB specification version number (e.g., 0200H for V2.00)
bDeviceClass	uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDeviceClass Class Code
bDeviceSubClass	uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDeviceSubClass SubClass Code
bDeviceProtocol	uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDeviceProtocol Protocol Code
bMaxPacketSize0	uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bMaxPacketSize0 Maximum packet size for other speed
bNumConfigurations	uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bNumConfigurations Number of Other-speed Configurations
bReserved	uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bReserved Reserved for future use, must be zero

34.4.19 _USB_DFU_FUNC_DESCRIPTOR

Table 476. _USB_DFU_FUNC_DESCRIPTOR class structure

Member	Description
bLength	uint8_t _USB_DFU_FUNC_DESCRIPTOR::bLength
bDescriptorType	uint8_t _USB_DFU_FUNC_DESCRIPTOR::bDescriptorType
bmAttributes	uint8_t _USB_DFU_FUNC_DESCRIPTOR::bmAttributes
wDetachTimeOut	uint16_t _USB_DFU_FUNC_DESCRIPTOR::wDetachTimeOut
wTransferSize	uint16_t _USB_DFU_FUNC_DESCRIPTOR::wTransferSize
bcdDFUVersion	uint16_t _USB_DFU_FUNC_DESCRIPTOR::bcdDFUVersion

34.4.20 _USB_INTERFACE_DESCRIPTOR

Table 477. _USB_INTERFACE_DESCRIPTOR class structure

Member	Description
bLength	uint8_t _USB_INTERFACE_DESCRIPTOR::bLength Size of this descriptor in bytes
bDescriptorType	uint8_t _USB_INTERFACE_DESCRIPTOR::bDescriptorType INTERFACE Descriptor Type
bInterfaceNumber	uint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceNumber Number of this interface. Zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
bAlternateSetting	uint8_t _USB_INTERFACE_DESCRIPTOR::bAlternateSetting Value used to select this alternate setting for the interface identified in the prior field
bNumEndpoints	uint8_t _USB_INTERFACE_DESCRIPTOR::bNumEndpoints Number of endpoints used by this interface (excluding endpoint zero). If this value is zero, this interface only uses the Default Control Pipe.
bInterfaceClass	uint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceClass Class code (assigned by the USB-IF).
bInterfaceSubClass	uint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceSubClass Subclass code (assigned by the USB-IF).
bInterfaceProtocol	uint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceProtocol Protocol code (assigned by the USB).
iInterface	uint8_t _USB_INTERFACE_DESCRIPTOR::iInterface Index of string descriptor describing this interface

34.4.21 _USB_OTHER_SPEED_CONFIGURATION

Table 478. _USB_OTHER_SPEED_CONFIGURATION class structure

Member	Description
bLength	uint8_t _USB_OTHER_SPEED_CONFIGURATION::bLength Size of descriptor
bDescriptorType	uint8_t _USB_OTHER_SPEED_CONFIGURATION::bDescriptorType Other_speed_Configuration Type
wTotalLength	uint16_t _USB_OTHER_SPEED_CONFIGURATION::wTotalLength Total length of data returned
bNumInterfaces	uint8_t _USB_OTHER_SPEED_CONFIGURATION::bNumInterfaces Number of interfaces supported by this speed configuration
bConfigurationValue	uint8_t _USB_OTHER_SPEED_CONFIGURATION::bConfigurationValue Value to use to select configuration
IConfiguration	uint8_t _USB_OTHER_SPEED_CONFIGURATION::IConfiguration Index of string descriptor
bmAttributes	uint8_t _USB_OTHER_SPEED_CONFIGURATION::bmAttributes Same as Configuration descriptor
bMaxPower	uint8_t _USB_OTHER_SPEED_CONFIGURATION::bMaxPower Same as Configuration descriptor

34.4.22 _USB_SETUP_PACKET

Table 479. _USB_SETUP_PACKET class structure

Member	Description
bmRequestType	REQUEST_TYPE _USB_SETUP_PACKET::bmRequestType This bit-mapped field identifies the characteristics of the specific request. _BM_T.
bRequest	uint8_t _USB_SETUP_PACKET::bRequest This field specifies the particular request. The Type bits in the bmRequestType field modify the meaning of this field. USBD_REQUEST.
wValue	WORD_BYTE _USB_SETUP_PACKET::wValue Used to pass a parameter to the device, specific to the request.
wIndex	WORD_BYTE _USB_SETUP_PACKET::wIndex Used to pass a parameter to the device, specific to the request. The wIndex field is often used in requests to specify an endpoint or an interface.
wLength	uint16_t _USB_SETUP_PACKET::wLength This field specifies the length of the data transferred during the second phase of the control transfer.

34.4.23 _USB_STRING_DESCRIPTOR

Table 480. _USB_STRING_DESCRIPTOR class structure

Member	Description
bLength	uint8_t _USB_STRING_DESCRIPTOR::bLength Size of this descriptor in bytes
bDescriptorType	uint8_t _USB_STRING_DESCRIPTOR::bDescriptorType STRING Descriptor Type
bString	uint16_t _USB_STRING_DESCRIPTOR::bString UNICODE encoded string

34.4.24 _WB_T

Table 481. _WB_T class structure

Member	Description
L	uint8_t _WB_T::L lower byte
H	uint8_t _WB_T::H upper byte

34.4.25 USBD_API

Main USBD API functions structure.This structure contains pointer to various USB Device stack's sub-module function tables. This structure is used as main entry point to access various methods (grouped in sub-modules) exposed by ROM based USB device stack.

Table 482. USBD_API class structure

Member	Description
hw	`const USBD_HW_API_T* USBD_API::hw` Pointer to function table which exposes functions which interact directly with USB device stack's core layer.
core	`const USBD_CORE_API_T* USBD_API::core` Pointer to function table which exposes functions which interact directly with USB device controller hardware.
msc	`const USBD_MSC_API_T* USBD_API::msc` Pointer to function table which exposes functions provided by MSC function driver module.
dfu	`const USBD_DFU_API_T* USBD_API::dfu` Pointer to function table which exposes functions provided by DFU function driver module.
hid	`const USBD_HID_API_T* USBD_API::hid` Pointer to function table which exposes functions provided by HID function driver module.
cdc	`const USBD_CDC_API_T* USBD_API::cdc` Pointer to function table which exposes functions provided by CDC-ACM function driver module.
reserved6	`const uint32_t* USBD_API::reserved6` Reserved for future function driver module.
version	`const uint32_t USBD_API::version` Version identifier of USB ROM stack. The version is defined as 0x0CHDMhCC where each nibble represents version number of the corresponding component. CC - 7:0 - 8bit core version number h - 11:8 - 4bit hardware interface version number M - 15:12 - 4bit MSC class module version number D - 19:16 - 4bit DFU class module version number H - 23:20 - 4bit HID class module version number C - 27:24 - 4bit CDC class module version number H - 31:28 - 4bit reserved

34.4.26 USBD_API_INIT_PARAM

USB device stack initialization parameter data structure.

Table 483. USBD_API_INIT_PARAM class structure

Member	Description
usb_reg_base	`uint32_t USBD_API_INIT_PARAM::usb_reg_base` USB device controller's base register address.
mem_base	`uint32_t USBD_API_INIT_PARAM::mem_base` Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 2048 byte boundary.
mem_size	`uint32_t USBD_API_INIT_PARAM::mem_size` The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_HW_API::GetMemSize() routine.

Table 483. USBD_API_INIT_PARAM class structure

Member	Description
max_num_ep	`uint8_t USBD_API_INIT_PARAM::max_num_ep` max number of endpoints supported by the USB device controller instance (specified by
pad0	`uint8_t USBD_API_INIT_PARAM::pad0[3][3]`
USB_Reset_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Reset_Event` Event for USB interface reset. This event fires when the USB host requests that the device reset its interface. This event fires after the control endpoint has been automatically configured by the library. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will prevent the device from enumerating correctly or operate properly.
USB_Suspend_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Suspend_Event` Event for USB suspend. This event fires when the USB host suspends the device by halting its transmission of Start Of Frame pulses to the device. This is generally hooked in order to move the device over to a low power state until the host wakes up the device. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will cause other system issues.
USB_Resume_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Resume_Event` Event for USB wake up or resume. This event fires when a the USB device interface is suspended and the host wakes up the device by supplying Start Of Frame pulses. This is generally hooked to pull the user application out of a low power state and back into normal operating mode. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will cause other system issues.
reserved_sbz	`USB_CB_T USBD_API_INIT_PARAM::reserved_sbz` Reserved parameter should be set to zero.
USB_SOF_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_SOF_Event` Event for USB Start Of Frame detection, when enabled. This event fires at the start of each USB frame, once per millisecond in full-speed mode or once per 125 microseconds in high-speed mode, and is synchronized to the USB bus. This event is time-critical; it is run once per millisecond (full-speed mode) and thus long handlers will significantly degrade device performance. This event should only be enabled when needed to reduce device wake-ups. This event is not normally active - it must be manually enabled and disabled via the USB interrupt register. **Remark:** This event is not normally active - it must be manually enabled and disabled via the USB interrupt register.

Table 483. USBD_API_INIT_PARAM class structure

Member	Description
USB_WakeUpCfg	`USB_PARAM_CB_T USBD_API_INIT_PARAM::USB_WakeUpCfg` Event for remote wake-up configuration, when enabled. This event fires when the USB host request the device to configure itself for remote wake-up capability. The USB host sends this request to device which report remote wake-up capable in their device descriptors, before going to low-power state. The application layer should implement this callback if they have any special on board circuit to triger remote wake up event. Also application can use this callback to differentiate the following SUSPEND event is caused by cable plug-out or host SUSPEND request. The device can wake-up host only after receiving this callback and remote wake-up feature is enabled by host. To signal remote wake-up the device has to generate resume signaling on bus by calling usapi.hw->WakeUp() routine. Parameters: 1. hUsb = Handle to the USB device stack. 2. param1 = When 0 - Clear the wake-up configuration, 1 - Enable the wake-up configuration. Returns: The call back should return ErrorCode_t type to indicate success or error condition.
USB_Power_Event	`USB_PARAM_CB_T USBD_API_INIT_PARAM::USB_Power_Event` Reserved parameter should be set to zero.
USB_Error_Event	`USB_PARAM_CB_T USBD_API_INIT_PARAM:USB_Error_Event` Event for error condition. This event fires when USB device controller detect an error condition in the system. Parameters: 1. hUsb = Handle to the USB device stack. 2. param1 = USB device interrupt status register. Returns: The call back should return ErrorCode_t type to indicate success or error condition.
USB_Configure_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Configure_Event` Event for USB configuration number changed. This event fires when a the USB host changes the selected configuration number. On receiving configuration change request from host, the stack enables/configures the endpoints needed by the new configuration before calling this callback function. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will prevent the device from enumerating correctly or operate properly.
USB_Interface_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Interface_Event` Event for USB interface setting changed. This event fires when a the USB host changes the interface setting to one of alternate interface settings. On receiving interface change request from host, the stack enables/configures the endpoints needed by the new alternate interface setting before calling this callback function. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will prevent the device from enumerating correctly or operate properly.

Table 483. USBD_API_INIT_PARAM class structure

Member	Description
USB_Feature_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Feature_Event` Event for USB feature changed. This event fires when a the USB host send set/clear feature request. The stack handles this request for USB_FEATURE_REMOTE_WAKEUP, USB_FEATURE_TEST_MODE and USB_FEATURE_ENDPOINT_STALL features only. On receiving feature request from host, the stack handle the request appropriately and then calls this callback function. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will prevent the device from enumerating correctly or operate properly.
virt_to_phys	`uint32_t(* USBD_API_INIT_PARAM::virt_to_phys)(void *vaddr)` Reserved parameter for future use. should be set to zero.
cache_flush	`void(* USBD_API_INIT_PARAM::cache_flush)(uint32_t *start_adr, uint32_t *end_adr)` Reserved parameter for future use. should be set to zero.

34.4.27 USBD_CDC_API

CDC class API functions structure.This module exposes functions which interact directly with USB device controller hardware.

Table 484. USBD_CDC_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_CDC_API::GetMemSize)(USBD_CDC_INIT_PARAM_T *param)` Function to determine the memory required by the CDC function driver module. This function is called by application layer before calling pUsbApi->CDC->Init(), to allocate memory used by CDC function driver module. The application should allocate the memory which is accessible by USB controller/DMA controller. **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing CDC function driver module initialization parameters. Returns: Returns the required memory size in bytes.

Table 484. USBD_CDC_API class structure

Member	Description
init	ErrorCode_t(*ErrorCode_t USBD_CDC_API::init)(USBD_HANDLE_T hUsb, USBD_CDC_INIT_PARAM_T *param, USBD_HANDLE_T *phCDC) Function to initialize CDC function driver module. This function is called by application layer to initialize CDC function driver module. hUsbHandle to the USB device stack. paramStructure containing CDC function driver module initialization parameters. Parameters: 1. hUsb = Handle to the USB device stack. 2. param = Structure containing CDC function driver module initialization parameters. Returns: Returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_USBD_BAD_MEM_BUF = Memory buffer passed is not 4-byte aligned or smaller than required. 3. ERR_API_INVALID_PARAM2 = Either CDC_Write() or CDC_Read() or CDC_Verify() callbacks are not defined. 4. ERR_USBD_BAD_INTF_DESC = Wrong interface descriptor is passed. 5. ERR_USBD_BAD_EP_DESC = Wrong endpoint descriptor is passed.
SendNotification	ErrorCode_t(*ErrorCode_t USBD_CDC_API::SendNotification)(USBD_HANDLE_T hCdc, uint8_t bNotification, uint16_t data) Function to send CDC class notifications to host. This function is called by application layer to send CDC class notifications to host. See usbcdc11.pdf, section 6.3, Table 67 for various notification types the CDC device can send. **Remark:** The current version of the driver only supports following notifications allowed by ACM subclass: CDC_NOTIFICATION_NETWORK_CONNECTION, CDC_RESPONSE_AVAILABLE, CDC_NOTIFICATION_SERIAL_STATE. For all other notifications application should construct the notification buffer appropriately and call hw->USB_WriteEP() for interrupt endpoint associated with the interface. Parameters: 1. hCdc = Handle to CDC function driver. 2. bNotification = Notification type allowed by ACM subclass. Should be CDC_NOTIFICATION_NETWORK_CONNECTION, CDC_RESPONSE_AVAILABLE or CDC_NOTIFICATION_SERIAL_STATE. For all other types ERR_API_INVALID_PARAM2 is returned. See usbcdc11.pdf, section 3.6.2.1, table 5. 3. data = Data associated with notification. For CDC_NOTIFICATION_NETWORK_CONNECTION a non-zero data value is interpreted as connected state. For CDC_RESPONSE_AVAILABLE this parameter is ignored. For CDC_NOTIFICATION_SERIAL_STATE the data should use bitmap values defined in usbcdc11.pdf, section 6.3.5, Table 69. Returns: Returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_API_INVALID_PARAM2 = If unsupported notification type is passed.

34.4.28 USBD_CDC_INIT_PARAM

Communication Device Class function driver initialization parameter data structure.

Table 485. USBD_CDC_INIT_PARAM class structure

Member	Description
mem_base	uint32_t USBD_CDC_INIT_PARAM::mem_base Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 4 byte boundary.
mem_size	uint32_t USBD_CDC_INIT_PARAM::mem_size The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_CDC_API::GetMemSize() routine.
cif_intf_desc	uint8_t * USBD_CDC_INIT_PARAM::cif_intf_desc Pointer to the control interface descriptor within the descriptor array
dif_intf_desc	uint8_t * USBD_CDC_INIT_PARAM::dif_intf_desc Pointer to the data interface descriptor within the descriptor array
CIC_GetRequest	ErrorCode_t(* USBD_CDC_INIT_PARAM::CIC_GetRequest)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t *length) Communication Interface Class specific get request call-back function. This function is provided by the application software. This function gets called when host sends CIC management element get requests. **Remark:** Applications implementing Abstract Control Model subclass can set this param to NULL. As the default driver parses ACM requests and calls the individual ACM call-back routines defined in this structure. For all other subclasses this routine should be provided by the application. The setup packet data (pSetup) is passed to the call-back so that application can extract the CIC request type and other associated data. By default the stack will assign pBuffer pointer to EP0Buff allocated at init. The application code can directly write data into this buffer as long as data is less than 64 byte. If more data has to be sent then application code should update pBuffer pointer and length accordingly. Parameters: 1. hCdc = Handle to CDC function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuffer = Pointer to a pointer of data buffer containing request data. Pointer-to-pointer is used to implement zero-copy buffers. See USBD_ZeroCopy for more details on zero-copy concept. 4. length = Amount of data to be sent back to host. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 485. USBD_CDC_INIT_PARAM class structure

Member	Description
CIC_SetRequest	ErrorCode_t(* USBD_CDC_INIT_PARAM::CIC_SetRequest)(USBD_HANDLE_T hCdc, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t length) Communication Interface Class specific set request call-back function. This function is provided by the application software. This function gets called when host sends a CIC management element requests. **Remark:** Applications implementing Abstract Control Model subclass can set this param to NULL. As the default driver parses ACM requests and calls the individual ACM call-back routines defined in this structure. For all other subclasses this routine should be provided by the application. The setup packet data (pSetup) is passed to the call-back so that application can extract the CIC request type and other associated data. If a set request has data associated, then this call-back is called twice. (1) First when setup request is received, at this time application code could update pBuffer pointer to point to the intended destination. The length param is set to 0 so that application code knows this is first time. By default the stack will assign pBuffer pointer to EP0Buff allocated at init. Note, if data length is greater than 64 bytes and application code doesn't update pBuffer pointer the stack will send STALL condition to host. (2) Second when the data is received from the host. This time the length param is set with number of data bytes received. Parameters: 1. hCdc = Handle to CDC function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuffer = Pointer to a pointer of data buffer containing request data. Pointer-to-pointer is used to implement zero-copy buffers. See USBD_ZeroCopy for more details on zero-copy concept. 4. length = Amount of data copied to destination buffer. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
CDC_BulkIN_Hdlr	ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_BulkIN_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event) Communication Device Class specific BULK IN endpoint handler. The application software should provide the BULK IN endpoint handler. Applications should transfer data depending on the communication protocol type set in descriptors. **Remark:** Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 485. USBD_CDC_INIT_PARAM class structure

Member	Description
CDC_BulkOUT_Hdlr	ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_BulkOUT_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event))(USBD_HANDLE_T hUsb, void *data, uint32_t event) Communication Device Class specific BULK OUT endpoint handler. The application software should provide the BULK OUT endpoint handler. Applications should transfer data depending on the communication protocol type set in descriptors. **Remark:** Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
SendEncpsCmd	ErrorCode_t(* USBD_CDC_INIT_PARAM::SendEncpsCmd)(USBD_HANDLE_T hCDC, uint8_t *buffer, uint16_t len) Abstract control model(ACM) subclass specific SEND_ENCAPSULATED_COMMAND request call-back function. This function is provided by the application software. This function gets called when host sends a SEND_ENCAPSULATED_COMMAND set request. Parameters: 1. hCdc = Handle to CDC function driver. 2. buffer = Pointer to the command buffer. 3. len = Length of the command buffer. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 485. USBD_CDC_INIT_PARAM class structure

Member	Description
GetEncpsResp	`ErrorCode_t(* USBD_CDC_INIT_PARAM::GetEncpsResp)(USBD_HANDLE_T hCDC, uint8_t **buffer, uint16_t *len)` Abstract control model(ACM) subclass specific GET_ENCAPSULATED_RESPONSE request call-back function. This function is provided by the application software. This function gets called when host sends a GET_ENCAPSULATED_RESPONSE request. Parameters: 1. hCdc = Handle to CDC function driver. 2. buffer = Pointer to a pointer of data buffer containing response data. Pointer-to-pointer is used to implement zero-copy buffers. See USBD_ZeroCopy for more details on zero-copy concept. 3. len = Amount of data to be sent back to host. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
SetCommFeature	`ErrorCode_t(* USBD_CDC_INIT_PARAM::SetCommFeature)(USBD_HANDLE_T hCDC, uint16_t feature, uint8_t *buffer, uint16_t len)` Abstract control model(ACM) subclass specific SET_COMM_FEATURE request call-back function. This function is provided by the application software. This function gets called when host sends a SET_COMM_FEATURE set request. Parameters: 1. hCdc = Handle to CDC function driver. 2. feature = Communication feature type. 3. buffer = Pointer to the settings buffer for the specified communication feature. 4. len = Length of the request buffer. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 485. USBD_CDC_INIT_PARAM class structure

Member	Description
GetCommFeature	ErrorCode_t(* USBD_CDC_INIT_PARAM::GetCommFeature)(USBD_HANDLE_T hCDC, uint16_t feature, uint8_t **pBuffer, uint16_t *len) Abstract control model(ACM) subclass specific GET_COMM_FEATURE request call-back function. This function is provided by the application software. This function gets called when host sends a GET_ENCAPSULATED_RESPONSE request. Parameters: 1. hCdc = Handle to CDC function driver. 2. feature = Communication feature type. 3. buffer = Pointer to a pointer of data buffer containing current settings for the communication feature. Pointer-to-pointer is used to implement zero-copy buffers. 4. len = Amount of data to be sent back to host. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 485. USBD_CDC_INIT_PARAM class structure

Member	Description
ClrCommFeature	ErrorCode_t(* USBD_CDC_INIT_PARAM::ClrCommFeature)(USBD_HANDLE_T hCDC, uint16_t feature) Abstract control model(ACM) subclass specific CLEAR_COMM_FEATURE request call-back function. This function is provided by the application software. This function gets called when host sends a CLEAR_COMM_FEATURE request. In the call-back the application should Clears the settings for a particular communication feature. Parameters: 1. hCdc = Handle to CDC function driver. 2. feature = Communication feature type. See usbcdc11.pdf, section 6.2.4, Table 47. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 485. USBD_CDC_INIT_PARAM class structure

Member	Description
SetCtrlLineState	`ErrorCode_t(* USBD_CDC_INIT_PARAM::SetCtrlLineState)(USBD_HANDLE_T hCDC, uint16_t state)` Abstract control model(ACM) subclass specific SET_CONTROL_LINE_STATE request call-back function. This function is provided by the application software. This function gets called when host sends a SET_CONTROL_LINE_STATE request. RS-232 signal used to tell the DCE device the DTE device is now present Parameters: 1. hCdc = Handle to CDC function driver. 2. state = The state value uses bitmap values defined the *USB CDC class specification document* published by usb.org. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
SendBreak	`ErrorCode_t(* USBD_CDC_INIT_PARAM::SendBreak)(USBD_HANDLE_T hCDC, uint16_t mstime)` Abstract control model(ACM) subclass specific SEND_BREAK request call-back function. This function is provided by the application software. This function gets called when host sends a SEND_BREAK request. Parameters: 1. hCdc = Handle to CDC function driver. 2. mstime = Duration of Break signal in milliseconds. If mstime is FFFFh, then the application should send break until another SendBreak request is received with the wValue of 0000h. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 485. USBD_CDC_INIT_PARAM class structure

Member	Description
SetLineCode	`ErrorCode_t(* USBD_CDC_INIT_PARAM::SetLineCode)(USBD_HANDLE_T hCDC, CDC_LINE_CODING *line_coding)` Abstract control model(ACM) subclass specific SET_LINE_CODING request call-back function. This function is provided by the application software. This function gets called when host sends a SET_LINE_CODING request. The application should configure the device per DTE rate, stop-bits, parity, and number-of-character bits settings provided in command buffer. Parameters: 1. hCdc = Handle to CDC function driver. 2. line_coding = Pointer to the CDC_LINE_CODING command buffer. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
CDC_InterruptEP_Hdlr	`ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_InterruptEP_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional Communication Device Class specific INTERRUPT IN endpoint handler. The application software should provide the INT IN endpoint handler. Applications should transfer data depending on the communication protocol type set in descriptors. Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 485. USBD_CDC_INIT_PARAM class structure

Member	Description
CDC_Ep0_Hdlr	`ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional user override-able function to replace the default CDC class handler. The application software could override the default EP0 class handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_CDC_API::Init(). Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

34.4.29 USBD_CORE_API

USBD stack Core API functions structure.

Table 486. USBD_CORE_API class structure

Member	Description
RegisterClassHandler	`ErrorCode_t(*ErrorCode_t USBD_CORE_API::RegisterClassHandler)(USBD_HANDLE_T hUsb, USB_EP_HANDLER_T pfn, void *data)` Function to register class specific EP0 event handler with USB device stack. The application layer uses this function when it has to register the custom class's EP0 handler. The stack calls all the registered class handlers on any EP0 event before going through default handling of the event. This gives the class handlers to implement class specific request handlers and also to override the default stack handling for a particular event targeted to the interface. Check USB_EP_HANDLER_T for more details on how the callback function should be implemented. Also application layer could use this function to register EP0 handler which responds to vendor specific requests. Parameters: 1. hUsb = Handle to the USB device stack. 2. pfn = Class specific EP0 handler function. 3. data = Pointer to the data which will be passed when callback function is called by the stack. Returns: Returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_USBD_TOO_MANY_CLASS_HDLR(0x0004000c) = The number of class handlers registered is greater than the number of handlers allowed by the stack.

Table 486. USBD_CORE_API class structure

Member	Description
RegisterEpHandler	`ErrorCode_t(*ErrorCode_t USBD_CORE_API::RegisterEpHandler)(USBD_HANDLE_T hUsb, uint32_t ep_index, USB_EP_HANDLER_T pfn, void *data)` Function to register interrupt/event handler for the requested endpoint with USB device stack. The application layer uses this function to register the custom class's EP0 handler. The stack calls all the registered class handlers on any EP0 event before going through default handling of the event. This gives the class handlers to implement class specific request handlers and also to override the default stack handling for a particular event targeted to the interface. Check USB_EP_HANDLER_T for more details on how the callback function should be implemented. Parameters: 1. hUsb = Handle to the USB device stack. 2. ep_index = Class specific EP0 handler function. 3. pfn = Class specific EP0 handler function. 4. data = Pointer to the data which will be passed when callback function is called by the stack. Returns: Returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_USBD_TOO_MANY_CLASS_HDLR(0x0004000c) = Too many endpoint handlers.
SetupStage	`void(*void USBD_CORE_API::SetupStage)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in setup state. This function is called by USB stack and the application layer to set the EP0 state machine in setup state. This function will read the setup packet received from USB host into stack's buffer. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: Nothing.
DataInStage	`void(*void USBD_CORE_API::DataInStage)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in data_in state. This function is called by USB stack and the application layer to set the EP0 state machine in data_in state. This function will write the data present in EP0Data buffer to EP0 FIFO for transmission to host. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: Nothing.

Table 486. USBD_CORE_API class structure

Member	Description
DataOutStage	void(*void USBD_CORE_API::DataOutStage)(USBD_HANDLE_T hUsb) Function to set EP0 state machine in data_out state. This function is called by USB stack and the application layer to set the EP0 state machine in data_out state. This function will read the control data (EP0 out packets) received from USB host into EP0Data buffer. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: Nothing.
StatusInStage	void(*void USBD_CORE_API::StatusInStage)(USBD_HANDLE_T hUsb) Function to set EP0 state machine in status_in state. This function is called by USB stack and the application layer to set the EP0 state machine in status_in state. This function will send zero length IN packet on EP0 to host, indicating positive status. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: Nothing.
StatusOutStage	void(*void USBD_CORE_API::StatusOutStage)(USBD_HANDLE_T hUsb) Function to set EP0 state machine in status_out state. This function is called by USB stack and the application layer to set the EP0 state machine in status_out state. This function will read the zero length OUT packet received from USB host on EP0. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: Nothing.

Table 486. USBD_CORE_API class structure

Member	Description
StallEp0	`void(*void USBD_CORE_API::StallEp0)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in stall state. This function is called by USB stack and the application layer to generate STALL signalling on EP0 endpoint. This function will also reset the EP0Data buffer. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: Nothing.

34.4.30 USBD_DFU_API

DFU class API functions structure.This module exposes functions which interact directly with USB device controller hardware.

Table 487. USBD_DFU_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_DFU_API::GetMemSize)(USBD_DFU_INIT_PARAM_T *param)` Function to determine the memory required by the DFU function driver module. This function is called by application layer before calling pUsbApi->dfu->Init(), to allocate memory used by DFU function driver module. The application should allocate the memory which is accessible by USB controller/DMA controller. **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing DFU function driver module initialization parameters. Returns: Returns the required memory size in bytes.
init	`ErrorCode_t(*ErrorCode_t USBD_DFU_API::init)(USBD_HANDLE_T hUsb, USBD_DFU_INIT_PARAM_T *param, uint32_t init_state)` Function to initialize DFU function driver module. This function is called by application layer to initialize DFU function driver module. Parameters: 1. hUsb = Handle to the USB device stack. 2. param = Structure containing DFU function driver module initialization parameters. Returns: Returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_USBD_BAD_MEM_BUF = Memory buffer passed is not 4-byte aligned or smaller than required. 3. ERR_API_INVALID_PARAM2 = Either DFU_Write() or DFU_Done() or DFU_Read() callbacks are not defined. 4. ERR_USBD_BAD_DESC = USB_DFU_DESCRIPTOR_TYPE is not defined immediately after interface descriptor.wTransferSize in descriptor doesn't match the value passed in param->wTransferSize.DFU_Detach() is not defined while USB_DFU_WILL_DETACH is set in DFU descriptor. 5. ERR_USBD_BAD_INTF_DESC = Wrong interface descriptor is passed.

34.4.31 USBD_DFU_INIT_PARAM

USB descriptors data structure.

Table 488. USBD_DFU_INIT_PARAM class structure

Member	Description
mem_base	`uint32_t USBD_DFU_INIT_PARAM::mem_base` Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 4 byte boundary.
mem_size	`uint32_t USBD_DFU_INIT_PARAM::mem_size` The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_DFU_API::GetMemSize() routine.

Table 488. USBD_DFU_INIT_PARAM class structure

Member	Description
wTransferSize	`uint16_t USBD_DFU_INIT_PARAM::wTransferSize` DFU transfer block size in number of bytes. This value should match the value set in DFU descriptor provided as part of the descriptor array (
pad	`uint16_t USBD_DFU_INIT_PARAM::pad`
intf_desc	`uint8_t * USBD_DFU_INIT_PARAM::intf_desc` Pointer to the DFU interface descriptor within the descriptor array (
DFU_Write	`uint8_t(*uint8_t(* USBD_DFU_INIT_PARAM::DFU_Write)(uint32_t block_num, uint8_t **src, uint32_t length, uint8_t *bwPollTimeout))(uint32_t block_num, uint8_t **src, uint32_t length, uint8_t *bwPollTimeout)` DFU Write callback function. This function is provided by the application software. This function gets called when host sends a write command. For application using zero-copy buffer scheme this function is called for the first time with Parameters: 1. block_num = Destination start address. 2. src = Pointer to a pointer to the source of data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 3. bwPollTimeout = Pointer to a 3 byte buffer which the callback implementer should fill with the amount of minimum time, in milliseconds, that the host should wait before sending a subsequent DFU_GETSTATUS request. 4. length = Number of bytes to be written. Returns: Returns DFU_STATUS_ values defined in mw_usbd_dfu.h.
DFU_Read	`uint32_t(*uint32_t(* USBD_DFU_INIT_PARAM::DFU_Read)(uint32_t block_num, uint8_t **dst, uint32_t length))(uint32_t block_num, uint8_t **dst, uint32_t length)` DFU Read callback function. This function is provided by the application software. This function gets called when host sends a read command. Parameters: 1. block_num = Destination start address. 2. dst = Pointer to a pointer to the source of data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 3. length = Amount of data copied to destination buffer. Returns: Returns DFU_STATUS_ values defined in mw_usbd_dfu.h.
DFU_Done	`void(*USBD_DFU_INIT_PARAM::DFU_Done)(void)` DFU done callback function. This function is provided by the application software. This function gets called after download is finished. Nothing. Returns: Nothing.

Table 488. USBD_DFU_INIT_PARAM class structure

Member	Description
DFU_Detach	void(* USBD_DFU_INIT_PARAM::DFU_Detach)(USBD_HANDLE_T hUsb) DFU detach callback function. This function is provided by the application software. This function gets called after USB_REQ_DFU_DETACH is received. Applications which set USB_DFU_WILL_DETACH bit in DFU descriptor should define this function. As part of this function application can call Connect() routine to disconnect and then connect back with host. For application which rely on WinUSB based host application should use this feature since USB reset can be invoked only by kernel drivers on Windows host. By implementing this feature host doesn't have to issue reset instead the device has to do it automatically by disconnect and connect procedure. hUsbHandle DFU control structure. Parameters: 1. hUsb = Handle DFU control structure. Returns: Nothing.
DFU_Ep0_Hdlr	ErrorCode_t(* USBD_DFU_INIT_PARAM::DFU_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event) Optional user overridable function to replace the default DFU class handler. The application software could override the default EP0 class handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_DFU_API::Init(). **Remark:** Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

34.4.32 USBD_HID_API

HID class API functions structure.This structure contains pointers to all the function exposed by HID function driver module.

Table 489. USBD_HID_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_HID_API::GetMemSize)(USBD_HID_INIT_PARAM_T *param)` Function to determine the memory required by the HID function driver module. This function is called by application layer before calling pUsbApi->hid->Init(), to allocate memory used by HID function driver module. The application should allocate the memory which is accessible by USB controller/DMA controller. **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing HID function driver module initialization parameters. Returns: Returns the required memory size in bytes.
init	`ErrorCode_t(*ErrorCode_t USBD_HID_API::init)(USBD_HANDLE_T hUsb, USBD_HID_INIT_PARAM_T *param)` Function to initialize HID function driver module. This function is called by application layer to initialize HID function driver module. On successful initialization the function returns a handle to HID function driver module in passed param structure. Parameters: 1. hUsb = Handle to the USB device stack. 2. param = Structure containing HID function driver module initialization parameters. Returns: Returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_USBD_BAD_MEM_BUF = Memory buffer passed is not 4-byte aligned or smaller than required. 3. ERR_API_INVALID_PARAM2 = Either HID_GetReport() or HID_SetReport() callback are not defined. 4. ERR_USBD_BAD_DESC = HID_HID_DESCRIPTOR_TYPE is not defined immediately after interface descriptor. 5. ERR_USBD_BAD_INTF_DESC = Wrong interface descriptor is passed. 6. ERR_USBD_BAD_EP_DESC = Wrong endpoint descriptor is passed.

34.4.33 USBD_HID_INIT_PARAM

USB descriptors data structure.

Table 490. USBD_HID_INIT_PARAM class structure

Member	Description
mem_base	uint32_t USBD_HID_INIT_PARAM::mem_base Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 4 byte boundary.
mem_size	uint32_t USBD_HID_INIT_PARAM::mem_size The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_HID_API::GetMemSize() routine.
max_reports	uint8_t USBD_HID_INIT_PARAM::max_reports Number of HID reports supported by this instance of HID class driver.
pad	uint8_t USBD_HID_INIT_PARAM::pad[3][3]
intf_desc	uint8_t * USBD_HID_INIT_PARAM::intf_desc Pointer to the HID interface descriptor within the descriptor array (
report_data	USB_HID_REPORT_T *USB_HID_REPORT_T* USBD_HID_INIT_PARAM::report_data Pointer to an array of HID report descriptor data structure (**Remark:** This array should be of global scope.
HID_GetReport	ErrorCode_t(* USBD_HID_INIT_PARAM::HID_GetReport)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t *length) HID get report callback function. This function is provided by the application software. This function gets called when host sends a HID_REQUEST_GET_REPORT request. The setup packet data (**Remark:** HID reports are sent via interrupt IN endpoint also. This function is called only when report request is received on control endpoint. Application should implement HID_EpIn_Hdlr to send reports to host via interrupt IN endpoint. Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuffer = Pointer to a pointer of data buffer containing report data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 4. length = Amount of data copied to destination buffer. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 490. USBD_HID_INIT_PARAM class structure

Member	Description
HID_SetReport	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_SetReport)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t length)` HID set report callback function. This function is provided by the application software. This function gets called when host sends a HID_REQUEST_SET_REPORT request. The setup packet data (Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuffer = Pointer to a pointer of data buffer containing report data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 4. length = Amount of data copied to destination buffer. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
HID_GetPhysDesc	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_GetPhysDesc)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuf, uint16_t *length)` Optional callback function to handle HID_GetPhysDesc request. The application software could provide this callback HID_GetPhysDesc handler to handle get physical descriptor requests sent by the host. When host requests Physical Descriptor set 0, application should return a special descriptor identifying the number of descriptor sets and their sizes. A Get_Descriptor request with the Physical Index equal to 1 should return the first Physical Descriptor set. A device could possibly have alternate uses for its items. These can be enumerated by issuing subsequent Get_Descriptor requests while incrementing the Descriptor Index. A device should return the last descriptor set to requests with an index greater than the last number defined in the HID descriptor. **Remark:** Applications which don't have physical descriptor should set this data member to zero before calling the USBD_HID_API::Init(). Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuf = Pointer to a pointer of data buffer containing physical descriptor data. If the physical descriptor is in USB accessible memory area application could just update the pointer or else it should copy the descriptor to the address pointed by this pointer. 4. length = Amount of data copied to destination buffer or descriptor length. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 490. USBD_HID_INIT_PARAM class structure

Member	Description
HID_SetIdle	ErrorCode_t(* USBD_HID_INIT_PARAM::HID_SetIdle)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t idleTime) Optional callback function to handle HID_REQUEST_SET_IDLE request. The application software could provide this callback to handle HID_REQUEST_SET_IDLE requests sent by the host. This callback is provided to applications to adjust timers associated with various reports, which are sent to host over interrupt endpoint. The setup packet data (**Remark:** Applications which don't send reports on Interrupt endpoint or don't have idle time between reports should set this data member to zero before calling the USBD_HID_API::Init(). Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet recived from host. 3. idleTime = Idle time to be set for the specified report. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
HID_SetProtocol	ErrorCode_t(* USBD_HID_INIT_PARAM::HID_SetProtocol)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t protocol) Optional callback function to handle HID_REQUEST_SET_PROTOCOL request. The application software could provide this callback to handle HID_REQUEST_SET_PROTOCOL requests sent by the host. This callback is provided to applications to adjust modes of their code between boot mode and report mode. **Remark:** Applications which don't support protocol modes should set this data member to zero before calling the USBD_HID_API::Init(). Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet recived from host. 3. protocol = Protocol mode. 0 = Boot Protocol 1 = Report Protocol Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 490. USBD_HID_INIT_PARAM class structure

Member	Description
HID_EpIn_Hdlr	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_EpIn_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional Interrupt IN endpoint event handler. The application software could provide Interrupt IN endpoint event handler. Application which send reports to host on interrupt endpoint should provide an endpoint event handler through this data member. This data member is ignored if the interface descriptor Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Handle to HID function driver. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: The call back should return ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 490. USBD_HID_INIT_PARAM class structure

Member	Description
HID_EpOut_Hdlr	ErrorCode_t(* USBD_HID_INIT_PARAM::HID_EpOut_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event) Optional Interrupt OUT endpoint event handler. The application software could provide Interrupt OUT endpoint event handler. Application which receives reports from host on interrupt endpoint should provide an endpoint event handler through this data member. This data member is ignored if the interface descriptor Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Handle to HID function driver. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: The call back should return ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
HID_GetReportDesc	ErrorCode_t(* USBD_HID_INIT_PARAM::HID_GetReportDesc)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuf, uint16_t *length) Optional user overridable function to replace the default HID_GetReportDesc handler. The application software could override the default HID_GetReportDesc handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_HID_API::Init() and also provide report data array **Remark:** Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 490. USBD_HID_INIT_PARAM class structure

Member	Description
HID_Ep0_Hdlr	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional user overridable function to replace the default HID class handler. The application software could override the default EP0 class handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_HID_API::Init(). **Remark:** Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

34.4.34 USBD_HW_API

Hardware API functions structure.This module exposes functions which interact directly with USB device controller hardware.

Table 491. USBD_HW_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_HW_API::GetMemSize)(USBD_API_INIT_PARAM_T *param)` Function to determine the memory required by the USB device stack's DCD and core layers. This function is called by application layer before calling pUsbApi->hw-> **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing USB device stack initialization parameters. Returns: Returns the required memory size in bytes.
Init	`ErrorCode_t(*ErrorCode_t USBD_HW_API::Init)(USBD_HANDLE_T *phUsb, USB_CORE_DESCS_T *pDesc, USBD_API_INIT_PARAM_T *param)` Function to initialize USB device stack's DCD and core layers. This function is called by application layer to initialize USB hardware and core layers. On successful initialization the function returns a handle to USB device stack which should be passed to the rest of the functions. Parameters: 1. phUsb = Pointer to the USB device stack handle of type USBD_HANDLE_T. 2. param = Structure containing USB device stack initialization parameters. Returns: Returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK(0) = On success 2. ERR_USBD_BAD_MEM_BUF(0x0004000b) = When insufficient memory buffer is passed or memory is not aligned on 2048 boundary.
Connect	`void(*void USBD_HW_API::Connect)(USBD_HANDLE_T hUsb, uint32_t con)` Function to make USB device visible/invisible on the USB bus. This function is called after the USB initialization. This function uses the soft connect feature to make the device visible on the USB bus. This function is called only after the application is ready to handle the USB data. The enumeration process is started by the host after the device detection. The driver handles the enumeration process according to the USB descriptors passed in the USB initialization function. Parameters: 1. hUsb = Handle to the USB device stack. 2. con = States whether to connect (1) or to disconnect (0). Returns: Nothing.

Table 491. USBD_HW_API class structure

Member	Description
ISR	void(*void USBD_HW_API::ISR)(USBD_HANDLE_T hUsb) Function to USB device controller interrupt events. When the user application is active the interrupt handlers are mapped in the user flash space. The user application must provide an interrupt handler for the USB interrupt and call this function in the interrupt handler routine. The driver interrupt handler takes appropriate action according to the data received on the USB bus. Parameters: 1. hUsb = Handle to the USB device stack. Returns: Nothing.
Reset	void(*void USBD_HW_API::Reset)(USBD_HANDLE_T hUsb) Function to Reset USB device stack and hardware controller. Reset USB device stack and hardware controller. Disables all endpoints except EP0. Clears all pending interrupts and resets endpoint transfer queues. This function is called internally by pUsbApi->hw->init() and from reset event. Parameters: 1. hUsb = Handle to the USB device stack. Returns: Nothing.
ForceFullSpeed	void(*void USBD_HW_API::ForceFullSpeed)(USBD_HANDLE_T hUsb, uint32_t cfg) Function to force high speed USB device to operate in full speed mode. This function is useful for testing the behavior of current device when connected to a full speed only hosts. Parameters: 1. hUsb = Handle to the USB device stack. 2. cfg = When 1 - set force full-speed or 0 - clear force full-speed. Returns: Nothing.

Table 491. USBD_HW_API class structure

Member	Description
WakeUpCfg	void(*void USBD_HW_API::WakeUpCfg)(USBD_HANDLE_T hUsb, uint32_t cfg) Function to configure USB device controller to walk-up host on remote events. This function is called by application layer to configure the USB device controller to wake up on remote events. It is recommended to call this function from users's USB_WakeUpCfg() callback routine registered with stack. **Remark:** User's USB_WakeUpCfg() is registered with stack by setting the USB_WakeUpCfg member of USBD_API_INIT_PARAM_T structure before calling pUsbApi->hw->Init() routine. Certain USB device controllers needed to keep some clocks always on to generate resume signaling through pUsbApi->hw->WakeUp(). This hook is provided to support such controllers. In most controllers cases this is an empty routine. Parameters: 1. hUsb = Handle to the USB device stack. 2. cfg = When 1 - Configure controller to wake on remote events or 0 - Configure controller not to wake on remote events. Returns: Nothing.
SetAddress	void(*void USBD_HW_API::SetAddress)(USBD_HANDLE_T hUsb, uint32_t adr) Function to set USB address assigned by host in device controller hardware. This function is called automatically when USB_REQUEST_SET_ADDRESS request is received by the stack from USB host. This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly. Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. 2. adr = USB bus Address to which the device controller should respond. Usually assigned by the USB host. Returns: Nothing.
Configure	void(*void USBD_HW_API::Configure)(USBD_HANDLE_T hUsb, uint32_t cfg) Function to configure device controller hardware with selected configuration. This function is called automatically when USB_REQUEST_SET_CONFIGURATION request is received by the stack from USB host. This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly. Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. 2. cfg = Configuration index. Returns: Nothing.

Table 491. USBD_HW_API class structure

Member	Description
ConfigEP	`void(*void USBD_HW_API::ConfigEP)(USBD_HANDLE_T hUsb, USB_ENDPOINT_DESCRIPTOR *pEPD)` Function to configure USB Endpoint according to descriptor. This function is called automatically when USB_REQUEST_SET_CONFIGURATION request is received by the stack from USB host. All the endpoints associated with the selected configuration are configured. This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly. Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. 2. pEPD = Endpoint descriptor structure defined in USB 2.0 specification. Returns: Nothing.
DirCtrlEP	`void(*void USBD_HW_API::DirCtrlEP)(USBD_HANDLE_T hUsb, uint32_t dir)` Function to set direction for USB control endpoint EP0. This function is called automatically by the stack on need basis. This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly. Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. 2. cfg = When 1 - Set EP0 in IN transfer mode 0 - Set EP0 in OUT transfer mode Returns: Nothing.

Table 491. USBD_HW_API class structure

Member	Description
EnableEP	`void(*void USBD_HW_API::EnableEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to enable selected USB endpoint. This function enables interrupts on selected endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: Nothing. This function enables interrupts on selected endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number corresponding to the event as per USB specification. ie. An EP1_IN is represented by 0x81 number. For device events set this param to 0x0. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. 4. enable = 1 - enable event, 0 - disable event. Returns: Returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK(0) = - On success 2. ERR_USBD_INVALID_REQ(0x00040001) = - Invalid event type.
DisableEP	`void(*void USBD_HW_API::DisableEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to disable selected USB endpoint. This function disables interrupts on selected endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: Nothing.
ResetEP	`void(*void USBD_HW_API::ResetEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to reset selected USB endpoint. This function flushes the endpoint buffers and resets data toggle logic. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: Nothing.

Table 491. USBD_HW_API class structure

Member	Description
SetStallEP	`void(*void USBD_HW_API::SetStallEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to STALL selected USB endpoint. Generates STALL signalling for requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: Nothing.
ClrStallEP	`void(*void USBD_HW_API::ClrStallEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to clear STALL state for the requested endpoint. This function clears STALL state for the requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: Nothing.
SetTestMode	`ErrorCode_t(*ErrorCode_t USBD_HW_API::SetTestMode)(USBD_HANDLE_T hUsb, uint8_t mode)` Function to set high speed USB device controller in requested test mode. USB-IF requires the high speed device to be put in various test modes for electrical testing. This USB device stack calls this function whenever it receives USB_REQUEST_CLEAR_FEATURE request for USB_FEATURE_TEST_MODE. Users can put the device in test mode by directly calling this function. Returns ERR_USBD_INVALID_REQ when device controller is full-speed only. Parameters: 1. hUsb = Handle to the USB device stack. 2. mode = Test mode defined in USB 2.0 electrical testing specification. Returns: Returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK(0) = - On success 2. ERR_USBD_INVALID_REQ(0x00040001) = - Invalid test mode or Device controller is full-speed only.

Table 491. USBD_HW_API class structure

Member	Description
ReadEP	`uint32_t(*uint32_t USBD_HW_API::ReadEP)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint8_t *pData)` Function to read data received on the requested endpoint. This function is called by USB stack and the application layer to read the data received on the requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. 3. pData = Pointer to the data buffer where data is to be copied. Returns: Returns the number of bytes copied to the buffer.
ReadReqEP	`uint32_t(*uint32_t USBD_HW_API::ReadReqEP)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint8_t *pData, uint32_t len)` Function to queue read request on the specified endpoint. This function is called by USB stack and the application layer to queue a read request on the specified endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. 3. pData = Pointer to the data buffer where data is to be copied. This buffer address should be accessible by USB DMA master. 4. len = Length of the buffer passed. Returns: Returns the length of the requested buffer.
ReadSetupPkt	`uint32_t(*uint32_t USBD_HW_API::ReadSetupPkt)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint32_t *pData)` Function to read setup packet data received on the requested endpoint. This function is called by USB stack and the application layer to read setup packet data received on the requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP0_IN is represented by 0x80 number. 3. pData = Pointer to the data buffer where data is to be copied. Returns: Returns the number of bytes copied to the buffer.

Table 491. USBD_HW_API class structure

Member	Description
WriteEP	`uint32_t(*uint32_t USBD_HW_API::WriteEP)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint8_t *pData, uint32_t cnt)` Function to write data to be sent on the requested endpoint. This function is called by USB stack and the application layer to send data on the requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. 3. pData = Pointer to the data buffer from where data is to be copied. 4. cnt = Number of bytes to write. Returns: Returns the number of bytes written.
WakeUp	`void(*void USBD_HW_API::WakeUp)(USBD_HANDLE_T hUsb)` Function to generate resume signaling on bus for remote host wake-up. This function is called by application layer to remotely wake up host controller when system is in suspend state. Application should indicate this remote wake up capability by setting USB_CONFIG_REMOTE_WAKEUP in bmAttributes of Configuration Descriptor. Also this routine will generate resume signalling only if host enables USB_FEATURE_REMOTE_WAKEUP by sending SET_FEATURE request before suspending the bus. Parameters: 1. hUsb = Handle to the USB device stack. Returns: Nothing.
EnableEvent	`ErrorCode_t(* USBD_HW_API::EnableEvent)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint32_t event_type, uint32_t enable)`

34.4.35 USBD_MSC_API

MSC class API functions structure.This module exposes functions which interact directly with USB device controller hardware.

Table 492. USBD_MSC_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_MSC_API::GetMemSize)(USBD_MSC_INIT_PARAM_T *param)` Function to determine the memory required by the MSC function driver module. This function is called by application layer before calling pUsbApi->msc->Init(), to allocate memory used by MSC function driver module. The application should allocate the memory which is accessible by USB controller/DMA controller. **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing MSC function driver module initialization parameters. Returns: Returns the required memory size in bytes.
init	`ErrorCode_t(*ErrorCode_t USBD_MSC_API::init)(USBD_HANDLE_T hUsb, USBD_MSC_INIT_PARAM_T *param)` Function to initialize MSC function driver module. This function is called by application layer to initialize MSC function driver module. Parameters: 1. hUsb = Handle to the USB device stack. 2. param = Structure containing MSC function driver module initialization parameters. Returns: Returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_USBD_BAD_MEM_BUF = Memory buffer passed is not 4-byte aligned or smaller than required. 3. ERR_API_INVALID_PARAM2 = Either MSC_Write() or MSC_Read() or MSC_Verify() callbacks are not defined. 4. ERR_USBD_BAD_INTF_DESC = Wrong interface descriptor is passed. 5. ERR_USBD_BAD_EP_DESC = Wrong endpoint descriptor is passed.

34.4.36 USBD_MSC_INIT_PARAM

Mass Storage class function driver initialization parameter data structure.

Table 493. USBD_MSC_INIT_PARAM class structure

Member	Description
mem_base	uint32_t USBD_MSC_INIT_PARAM::mem_base Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 4 byte boundary.
mem_size	uint32_t USBD_MSC_INIT_PARAM::mem_size The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_MSC_API::GetMemSize() routine.
InquiryStr	uint8_t * USBD_MSC_INIT_PARAM::InquiryStr Pointer to the 28 character string. This string is sent in response to the SCSI Inquiry command. **Remark:** The data pointed by the pointer should be of global scope.
BlockCount	uint32_t USBD_MSC_INIT_PARAM::BlockCount Number of blocks present in the mass storage device
BlockSize	uint32_t USBD_MSC_INIT_PARAM::BlockSize Block size in number of bytes
MemorySize	uint32_t USBD_MSC_INIT_PARAM::MemorySize Memory size in number of bytes
intf_desc	uint8_t * USBD_MSC_INIT_PARAM::intf_desc Pointer to the interface descriptor within the descriptor array (
MSC_Write	void(*void(* USBD_MSC_INIT_PARAM::MSC_Write)(uint32_t offset, uint8_t **src, uint32_t length))(uint32_t offset, uint8_t **src, uint32_t length) MSC Write callback function. This function is provided by the application software. This function gets called when host sends a write command. Parameters: 1. offset = Destination start address. 2. src = Pointer to a pointer to the source of data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 3. length = Number of bytes to be written. Returns: Nothing.

Table 493. USBD_MSC_INIT_PARAM class structure

Member	Description
MSC_Read	`void(*void(* USBD_MSC_INIT_PARAM::MSC_Read)(uint32_t offset, uint8_t **dst, uint32_t length))(uint32_t offset, uint8_t **dst, uint32_t length)` MSC Read callback function. This function is provided by the application software. This function gets called when host sends a read command. Parameters: 1. offset = Source start address. 2. dst = Pointer to a pointer to the source of data. The MSC function drivers implemented in stack are written with zero-copy model. Meaning the stack doesn't make an extra copy of buffer before writing/reading data from USB hardware FIFO. Hence the parameter is pointer to a pointer containing address buffer (uint8_t** dst). So that the user application can update the buffer pointer instead of copying data to address pointed by the parameter. /note The updated buffer address should be access able by USB DMA master. If user doesn't want to use zero-copy model, then the user should copy data to the address pointed by the passed buffer pointer parameter and shouldn't change the address value. See Zero-Copy Data Transfer model for more details on zero-copy concept. 3. length = Number of bytes to be read. Returns: Nothing.

Table 493. USBD_MSC_INIT_PARAM class structure

Member	Description
MSC_Verify	ErrorCode_t(* USBD_MSC_INIT_PARAM::MSC_Verify)(uint32_t offset, uint8_t buf[], uint32_t length) MSC Verify callback function. This function is provided by the application software. This function gets called when host sends a verify command. The callback function should compare the buffer with the destination memory at the requested offset and Parameters: 1. offset = Destination start address. 2. buf = Buffer containing the data sent by the host. 3. length = Number of bytes to verify. Returns: Returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = If data in the buffer matches the data at destination 2. ERR_FAILED = At least one byte is different.
MSC_GetWriteBuf	void(*void(* USBD_MSC_INIT_PARAM::MSC_GetWriteBuf)(uint32_t offset, uint8_t **buff_adr, uint32_t length))(uint32_t offset, uint8_t **buff_adr, uint32_t length) Optional callback function to optimize MSC_Write buffer transfer. This function is provided by the application software. This function gets called when host sends SCSI_WRITE10/SCSI_WRITE12 command. The callback function should update the Parameters: 1. offset = Destination start address. 2. buf = Buffer containing the data sent by the host. 3. length = Number of bytes to write. Returns: Nothing.

Table 493. USBD_MSC_INIT_PARAM class structure

Member	Description
MSC_Ep0_Hdlr	ErrorCode_t(* USBD_MSC_INIT_PARAM::MSC_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event) Optional user overridable function to replace the default MSC class handler. The application software could override the default EP0 class handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_MSC_API::Init(). **Remark:** Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

UM10732

Chapter 35: LPC11U6x/Ex Pin description

Rev. 1.8 — 19 December 2016 User manual

35.1 Pin description

35.1.1 LPC11U6x Pin description

Table 494. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART3 and USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100	Reset state[1]	Type	Description of pin functions
RESET/PIO0_0	3	4	8 [8]	I; PU	I	**RESET** — External reset input with 20 ns glitch filter. A LOW-going pulse as short as 50 ns on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0. This pin also serves as the debug select input. LOW level selects the JTAG boundary scan. HIGH level selects the ARM SWD debug mode. In deep power-down mode, this pin must be pulled HIGH externally. The RESET pin can be left unconnected or be used as a GPIO pin if an external RESET function is not needed and Deep power-down mode is not used.
					IO	**PIO0_0** — General-purpose digital input/output pin.
PIO0_1	4	5	9 [6]	I; PU	IO	**PIO0_1** — General-purpose digital input/output pin. A LOW level on this pin during reset starts the ISP command handler or the USB device enumeration.
					O	**CLKOUT** — Clockout pin.
					O	**CT32B0_MAT2** — Match output 2 for 32-bit timer 0.
					O	**USB_FTOGGLE** — USB 1 ms Start-of-Frame signal.
PIO0_2	11	14	19 [6]	I; PU	IO	**PIO0_2** — General-purpose port 0 input/output 2.
					IO	**SSP0_SSEL** — Slave select for SSP0.
					I	**CT16B0_CAP0** — Capture input 0 for 16-bit timer 0.
					-	**R_0** — Reserved.
PIO0_3	14	19	30 [6]	I; PU	IO	**PIO0_3** — General-purpose digital input/output pin. A LOW level on this pin during reset starts the ISP command handler. A HIGH level during reset starts the USB device enumeration.
					I	**USB_VBUS** — Monitors the presence of USB bus power.
					-	**R_1** — Reserved.
PIO0_4	15	20	31 [7]	IA	IO	**PIO0_4** — General-purpose port 0 input/output 4 (open-drain).
					IO	**I2C0_SCL** — I^{2}C-bus clock input/output (open-drain). High-current sink only if I^{2}C Fast-mode Plus is selected in the I/O configuration register.
					-	**R_2** — Reserved.

Table 494. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART3 and USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100	Reset state[1]	Type	Description of pin functions
PIO0_5	16	21	32 [7]	IA	IO	**PIO0_5 —** General-purpose port 0 input/output 5 (open-drain).
					IO	**I2C0_SDA —** I^{2}C-bus data input/output (open-drain). High-current sink only if I^{2}C Fast-mode Plus is selected in the I/O configuration register.
					-	**R_3 —** Reserved.
PIO0_6	23	29	44 [6]	I; PU	IO	**PIO0_6 —** General-purpose port 0 input/output 6.
					-	**R —** Reserved.
					IO	**SSP0_SCK —** Serial clock for SSP0.
					-	**R_4 —** Reserved.
PIO0_7	24	30	45 [5]	I; PU	IO	**PIO0_7 —** General-purpose port 0 input/output 7 (high-current output driver).
					I	**U0_$\overline{\text{CTS}}$ —** Clear To Send input for USART.
					-	**R_5 —** Reserved.
					IO	**I2C1_SCL —** I^{2}C-bus clock input/output. This pin is not open-drain.
PIO0_8	26	37	58 [6]	I; PU	IO	**PIO0_8 —** General-purpose port 0 input/output 8.
					IO	**SSP0_MISO —** Master In Slave Out for SSP0.
					O	**CT16B0_MAT0 —** Match output 0 for 16-bit timer 0.
					-	**R_6 —** Reserved.
PIO0_9	27	38	59 [6]	I; PU	IO	**PIO0_9 —** General-purpose port 0 input/output 9.
					IO	**SSP0_MOSI —** Master Out Slave In for SSP0.
					O	**CT16B0_MAT1 —** Match output 1 for 16-bit timer 0.
					-	**R_7 —** Reserved.
SWCLK/PIO0_10	28	39	60 [6]	I; PU	IO	**SWCLK —** Serial Wire Clock. SWCLK is enabled by default on this pin. In boundary scan mode: TCK (Test Clock).
					IO	**PIO0_10 —** General-purpose digital input/output pin.
					IO	**SSP0_SCK —** Serial clock for SSP0.
					O	**CT16B0_MAT2 —** 16-bit timer0 MAT2
TDI/PIO0_11	30	42	64 [3]	I; PU	IO	**TDI —** Test Data In for JTAG interface. In boundary scan mode only.
					IO	**PIO0_11 —** General-purpose digital input/output pin.
					AI	**ADC_9 —** A/D converter, input channel 9.
					O	**CT32B0_MAT3 —** Match output 3 for 32-bit timer 0.
					O	**U1_$\overline{\text{RTS}}$ —** Request To Send output for USART1.
					IO	**U1_SCLK —** Serial clock input/output for USART1 in synchronous mode.

Table 494. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART3 and USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100		Reset state[1]	Type	Description of pin functions
TMS/PIO0_12	31	43	66	[3]	I; PU	IO	**TMS —** Test Mode Select for JTAG interface. In boundary scan mode only.
						IO	**PIO0_12 —** General-purpose digital input/output pin.
						AI	**ADC_8 —** A/D converter, input channel 8.
						I	**CT32B1_CAP0 —** Capture input 0 for 32-bit timer 1.
						I	**U1_$\overline{\text{CTS}}$ —** Clear To Send input for USART1.
TDO/PIO0_13	32	45	68	[3]	I; PU	IO	**TDO —** Test Data Out for JTAG interface. In boundary scan mode only.
						IO	**PIO0_13 —** General-purpose digital input/output pin.
						AI	**ADC_7 —** A/D converter, input channel 7.
						O	**CT32B1_MAT0 —** Match output 0 for 32-bit timer 1.
						I	**U1_RXD —** Receiver input for USART1.
$\overline{\text{TRST}}$/PIO0_14	33	46	69	[3]	I; PU	IO	**$\overline{\text{TRST}}$ —** Test Reset for JTAG interface. In boundary scan mode only.
						IO	**PIO0_14 —** General-purpose digital input/output pin.
						AI	**ADC_6 —** A/D converter, input channel 6.
						O	**CT32B1_MAT1 —** Match output 1 for 32-bit timer 1.
						O	**U1_TXD —** Transmitter output for USART1.
SWDIO/PIO0_15	37	50	81	[3]	I; PU	IO	**SWDIO —** Serial Wire Debug I/O. SWDIO is enabled by default on this pin. In boundary scan mode: TMS (Test Mode Select).
						IO	**PIO0_15 —** General-purpose digital input/output pin.
						AI	**ADC_3 —** A/D converter, input channel 3.
						O	**CT32B1_MAT2 —** Match output 2 for 32-bit timer 1.
PIO0_16/WAKEUP	38	51	82	[4]	I; PU	IO	**PIO0_16 —** General-purpose digital input/output pin. This pin also serves as the Deep power-down mode wake-up pin with 20 ns glitch filter. Pull this pin HIGH externally before entering 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.
						AI	**ADC_2 —** A/D converter, input channel 2.
						O	**CT32B1_MAT3 —** Match output 3 for 32-bit timer 1.
						-	**R_8 —** Reserved.
PIO0_17	42	56	90	[6]	I; PU	IO	**PIO0_17 —** General-purpose digital input/output pin.
						O	**U0_$\overline{\text{RTS}}$ —** Request To Send output for USART0.
						I	**CT32B0_CAP0 —** Capture input 0 for 32-bit timer 0.
						IO	**U0_SCLK —** Serial clock input/output for USART0 in synchronous mode.

Table 494. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART3 and USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100	Reset state[1]	Type	Description of pin functions
PIO0_18	45	60	94 [6]	I; PU	IO	**PIO0_18 —** General-purpose digital input/output pin.
					I	**U0_RXD —** Receiver input for USART0. Used in UART ISP mode.
					O	**CT32B0_MAT0 —** Match output 0 for 32-bit timer 0.
PIO0_19	46	61	95 [6]	I; PU	IO	**PIO0_19 —** General-purpose digital input/output pin.
					O	**U0_TXD —** Transmitter output for USART0. Used in UART ISP mode.
					O	**CT32B0_MAT1 —** Match output 1 for 32-bit timer 0.
PIO0_20	10	12	17 [6]	I; PU	IO	**PIO0_20 —** General-purpose digital input/output pin.
					I	**CT16B1_CAP0 —** Capture input 0 for 16-bit timer 1.
					I	**U2_RXD —** Receiver input for USART2.
PIO0_21	17	22	33 [6]	I; PU	IO	**PIO0_21 —** General-purpose digital input/output pin.
					O	**CT16B1_MAT0 —** Match output 0 for 16-bit timer 1.
					IO	**SSP1_MOSI —** Master Out Slave In for SSP1.
PIO0_22	29	40	62 [3]	I; PU	IO	**PIO0_22 —** General-purpose digital input/output pin.
					AI	**ADC_11 —** A/D converter, input channel 11.
					I	**CT16B1_CAP1 —** Capture input 1 for 16-bit timer 1.
					IO	**SSP1_MISO —** Master In Slave Out for SSP1.
PIO0_23	39	52	83 [3]	I; PU	IO	**PIO0_23 —** General-purpose digital input/output pin.
					AI	**ADC_1 —** A/D converter, input channel 1.
					-	**R_9 —** Reserved.
					I	**U0_RI —** Ring Indicator input for USART0.
					IO	**SSP1_SSEL —** Slave select for SSP1.
PIO1_0	-	62	97 [6]	I; PU	IO	**PIO1_0 —** General-purpose digital input/output pin.
					O	**CT32B1_MAT0 —** Match output 0 for 32-bit timer 1.
					-	**R_10 —** Reserved.
					O	**U2_TXD —** Transmitter output for USART2.
PIO1_1	-	-	28 [6]	I; PU	IO	**PIO1_1 —** General-purpose digital input/output pin.
					O	**CT32B1_MAT1 —** Match output 1 for 32-bit timer 1.
					-	**R_11 —** Reserved.
					O	**U0_DTR —** Data Terminal Ready output for USART0.
PIO1_2	-	-	55 [6]	I; PU	IO	**PIO1_2 —** General-purpose digital input/output pin.
					O	**CT32B1_MAT2 —** Match output 2 for 32-bit timer 1.
					-	**R_12 —** Reserved.
					I	**U1_RXD —** Receiver input for USART1.

Table 494. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART3 and USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100	Reset state[1]	Type	Description of pin functions
PIO1_3	-	-	72 [3]	I; PU	IO	**PIO1_3 —** General-purpose digital input/output pin.
					O	**CT32B1_MAT3 —** Match output 3 for 32-bit timer 1.
					-	**R_13 —** Reserved.
					IO	**I2C1_SDA —** I^2C-bus data input/output (not open-drain).
					AI	**ADC_5 —** A/D converter, input channel 5.
PIO1_4	-	-	23 [6]	I; PU	IO	**PIO1_4 —** General-purpose digital input/output pin.
					I	**CT32B1_CAP0 —** Capture input 0 for 32-bit timer 1.
					-	**R_14 —** Reserved.
					I	**$\overline{\text{U0\_DSR}}$ —** Data Set Ready input for USART0.
PIO1_5	-	-	47 [6]	I; PU	IO	**PIO1_5 —** General-purpose digital input/output pin.
					I	**CT32B1_CAP1 —** Capture input 1 for 32-bit timer 1.
					-	**R_15 —** Reserved.
					I	**$\overline{\text{U0\_DCD}}$ —** Data Carrier Detect input for USART0.
PIO1_6	-	-	98 [6]	I; PU	IO	**PIO1_6 —** General-purpose digital input/output pin.
					-	**R_16 —** Reserved.
					I	**U2_RXD —** Receiver input for USART2.
					I	**CT32B0_CAP2 —** Capture input 2 for 32-bit timer 0.
PIO1_7	-	6	10 [6]	I; PU	IO	**PIO1_7 —** General-purpose digital input/output pin.
					-	**R_17 —** Reserved.
					I	**$\overline{\text{U2\_CTS}}$ —** Clear To Send input for USART2.
					I	**CT16B1_CAP0 —** Capture input 0 for 32-bit timer 1.
PIO1_8	-	-	61 [6]	I; PU	IO	**PIO1_8 —** General-purpose digital input/output pin.
					-	**R_18 —** Reserved.
					O	**U1_TXD —** Transmitter output for USART1.
					I	**CT16B0_CAP0 —** Capture input 0 for 16-bit timer 0.
PIO1_9	-	55	86 [3]	I; PU	IO	**PIO1_9 —** General-purpose digital input/output pin.
					I	**$\overline{\text{U0\_CTS}}$ —** Clear To Send input for USART0.
					O	**CT16B1_MAT1 —** Match output 1 for 16-bit timer 1.
					I	**ADC_0 —** A/D converter, input channel 0.
PIO1_10	-	13	18 [6]	I; PU	IO	**PIO1_10 —** General-purpose digital input/output pin.
					O	**$\overline{\text{U2\_RTS}}$ —** Request To Send output for USART2.
					IO	**U2_SCLK —** Serial clock input/output for USART2 in synchronous mode.
					O	**CT16B1_MAT0 —** Match output 0 for 16-bit timer 1.
PIO1_11	-	-	65 [6]	I; PU	IO	**PIO1_11 —** General-purpose digital input/output pin.
					IO	**I2C1_SCL —** I^2C1-bus clock input/output (not open-drain).
					O	**CT16B0_MAT2 —** Match output 2 for 16-bit timer 0.
					I	**$\overline{\text{U0\_RI}}$ —** Ring Indicator input for USART0.

Table 494. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART3 and USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100	Reset state[1]	Type	Description of pin functions
PIO1_12	-	-	89 [6]	I; PU	IO	**PIO1_12 —** General-purpose digital input/output pin.
					IO	**SSP0_MOSI —** Master Out Slave In for SSP0.
					O	**CT16B0_MAT1 —** Match output 1 for 16-bit timer 0.
					-	**R_21 —** Reserved.
PIO1_13	36	49	78 [6]	I; PU	IO	**PIO1_13 —** General-purpose digital input/output pin.
					I	**$\overline{\text{U1\_CTS}}$ —** Clear To Send input for USART1.
					O	**SCT0_OUT3 —** SCTimer0/PWM output 3.
					-	**R_22 —** Reserved.
PIO1_14	-	-	79 [6]	I; PU	IO	**PIO1_14 —** General-purpose digital input/output pin.
					IO	**I2C1_SDA —** I^{2}C1-bus data input/output (not open-drain).
					O	**CT32B1_MAT2 —** Match output 2 for 32-bit timer 1.
					-	**R_23 —** Reserved.
PIO1_15	-	-	87 [6]	I; PU	IO	**PIO1_15 —** General-purpose digital input/output pin.
					IO	**SSP0_SSEL —** Slave select for SSP0.
					O	**CT32B1_MAT3 —** Match output 3 for 32-bit timer 1.
					-	**R_24 —** Reserved.
PIO1_16	-	-	96 [6]	I; PU	IO	**PIO1_16 —** General-purpose digital input/output pin.
					IO	**SSP0_MISO —** Master In Slave Out for SSP0.
					O	**CT16B0_MAT0 —** Match output 0 for 16-bit timer 0.
					-	**R_25 —** Reserved.
PIO1_17	-	-	34 [6]	I; PU	IO	**PIO1_17 —** General-purpose digital input/output pin.
					I	**CT16B0_CAP2 —** Capture input 2 for 16-bit timer 0.
					I	**U0_RXD —** Receiver input for USART0.
					-	**R_26 —** Reserved.
PIO1_18	-	-	43 [6]	I; PU	IO	**PIO1_18 —** General-purpose digital input/output pin.
					I	**CT16B1_CAP1 —** Capture input 1 for 16-bit timer 1.
					O	**U0_TXD —** Transmitter output for USART0.
					-	**R_27 —** Reserved.
PIO1_19	-	64	4 [6]	I; PU	IO	**PIO1_19 —** General-purpose digital input/output pin.
					I	**$\overline{\text{U2\_CTS}}$ —** Clear To Send input for USART2.
					O	**SCT0_OUT0 —** SCTimer0/PWM output 0.
					-	**R_28 —** Reserved.
PIO1_20	13	18	29 [6]	I; PU	IO	**PIO1_20 —** General-purpose digital input/output pin.
					I	**$\overline{\text{U0\_DSR}}$ —** Data Set Ready input for USART0.
					IO	**SSP1_SCK —** Serial clock for SSP1.
					O	**CT16B0_MAT0 —** Match output 0 for 16-bit timer 0.

Table 494. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART3 and USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100	Reset state[1]	Type	Description of pin functions
PIO1_21	25	35	56 [6]	I; PU	IO	**PIO1_21** — General-purpose digital input/output pin.
					I	**U0_$\overline{\text{DCD}}$** — Data Carrier Detect input for USART0.
					IO	**SSP1_MISO** — Master In Slave Out for SSP1.
					I	**CT16B0_CAP2** — Capture input 2 for 16-bit timer 0.
PIO1_22	-	-	80 [3]	I; PU	IO	**PIO1_22** — General-purpose digital input/output pin.
					IO	**SSP1_MOSI** — Master Out Slave In for SSP1.
					I	**CT32B1_CAP1** — Capture input 1 for 32-bit timer 1.
					AI	**ADC_4** — A/D converter, input channel 4.
					-	**R_29** — Reserved.
PIO1_23	18	23	35 [6]	I; PU	IO	**PIO1_23** — General-purpose digital input/output pin.
					O	**CT16B1_MAT1** — Match output 1 for 16-bit timer 1.
					IO	**SSP1_SSEL** — Slave select for SSP1.
					O	**U2_TXD** — Transmitter output for USART2.
PIO1_24	22	28	42 [6]	I; PU	IO	**PIO1_24** — General-purpose digital input/output pin.
					O	**CT32B0_MAT0** — Match output 0 for 32-bit timer 0.
					IO	**I2C1_SDA** — I^{2}C-bus data input/output (not open-drain).
PIO1_25	-	-	100 [6]	I; PU	IO	**PIO1_25** — General-purpose digital input/output pin.
					O	**U2_$\overline{\text{RTS}}$** — Request To Send output for USART2.
					IO	**U2_SCLK** — Serial clock input/output for USART2 in synchronous mode.
					I	**SCT0_IN0** — SCTimer0/PWM input 0.
					-	**R_30** — Reserved.
PIO1_26	-	15	20 [6]	I; PU	IO	**PIO1_26** — General-purpose digital input/output pin.
					O	**CT32B0_MAT2** — Match output 2 for 32-bit timer 0.
					I	**U0_RXD** — Receiver input for USART0.
					-	**R_19** — Reserved.
PIO1_27	-	17	22 [6]	I; PU	IO	**PIO1_27** — General-purpose digital input/output pin.
					O	**CT32B0_MAT3** — Match output 3 for 32-bit timer 0.
					O	**U0_TXD** — Transmitter output for USART0.
					-	**R_20** — Reserved.
					IO	**SSP1_SCK** — Serial clock for SSP1.
PIO1_28	-	31	46 [6]	I; PU	IO	**PIO1_28** — General-purpose digital input/output pin.
					I	**CT32B0_CAP0** — Capture input 0 for 32-bit timer 0.
					IO	**U0_SCLK** — Serial clock input/output for USART in synchronous mode.
					O	**U0_$\overline{\text{RTS}}$** — Request To Send output for USART0.

Table 494. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART3 and USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100		Reset state[1]	Type	Description of pin functions
PIO1_29	-	41	63	[3]	I; PU	IO	**PIO1_29 —** General-purpose digital input/output pin.
						IO	**SSP0_SCK —** Serial clock for SSP0.
						I	**CT32B0_CAP2 —** Capture input 2 for 32-bit timer 0.
						O	**U0_DTR —** Data Terminal Ready output for USART0.
						AI	**ADC_10 —** A/D converter, input channel 10.
PIO1_30	-	44	67	[6]	I; PU	IO	**PIO1_30 —** General-purpose digital input/output pin.
						IO	**I2C1_SCL —** I^{2}C1-bus clock input/output (not open-drain).
						I	**SCT0_IN3 —** SCTimer0/PWM input 3.
						-	**R_31 —** Reserved.
PIO1_31	-	-	48	[5]	I; PU	IO	**PIO1_31 —** General-purpose digital input/output pin (high-current output driver).
PIO2_0	6	8	12	[10]	I; PU	IO	**PIO2_0 —** General-purpose digital input/output pin.
						AI	**XTALIN —** Input to the oscillator circuit and internal clock generator circuits. Input voltage must not exceed 1.8 V.
PIO2_1	7	9	13	[10]	I; PU	IO	**PIO2_1 —** General-purpose digital input/output pin.
						AO	**XTALOUT —** Output from the oscillator amplifier.
PIO2_2	12	16	21	[6]	I; PU	IO	**PIO2_2 —** General-purpose digital input/output pin.
						O	**U3_RTS —** Request To Send output for USART3.
						IO	**U3_SCLK —** Serial clock input/output for USART3 in synchronous mode.
							SCT0_OUT1 — SCTimer0/PWM output 1.
PIO2_3	-	-	36	[6]	I; PU	IO	**PIO2_3 —** General-purpose digital input/output pin.
						I	**U3_RXD —** Receiver input for USART3.
						O	**CT32B0_MAT1 —** Match output 1 for 32-bit timer 0.
PIO2_4	-	-	41	[6]	I; PU	IO	**PIO2_4 —** General-purpose digital input/output pin.
						O	**U3_TXD —** Transmitter output for USART3.
						O	**CT32B0_MAT2 —** Match output 2 for 32-bit timer 0.
PIO2_5	9	11	15	[6]	I; PU	IO	**PIO2_5 —** General-purpose digital input/output pin.
						I	**U3_CTS —** Clear To Send input for USART3.
						I	**SCT0_IN1 —** SCTimer0/PWM input 1.
PIO2_6	-	24	37	[6]	I; PU	IO	**PIO2_6 —** General-purpose digital input/output pin.
						O	**U1_RTS —** Request To Send output for USART1.
						IO	**U1_SCLK —** Serial clock input/output for USART1 in synchronous mode.
						I	**SCT0_IN2 —** SCTimer0/PWM input 2.
PIO2_7	21	27	40	[6]	I; PU	IO	**PIO2_7 —** General-purpose digital input/output pin.
						IO	**SSP0_SCK —** Serial clock for SSP0.
						I	**SCT0_OUT2 —** SCTimer0/PWM output 2.

Table 494. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART3 and USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100	Reset state[1]	Type	Description of pin functions
PIO2_8	-	-	2 [6]	I; PU	IO	**PIO2_8 —** General-purpose digital input/output pin.
					I	**SCT1_IN0 —** SCTimer1/PWM input 0.
PIO2_9	-	-	3 [6]	I; PU	IO	**PIO2_9 —** General-purpose digital input/output pin.
					I	**SCT1_IN1 —** SCTimer1/PWM_IN1
PIO2_10	-	-	16 [6]	I; PU	IO	**PIO2_10 —** General-purpose digital input/output pin.
					O	**U4_RTS —** Request To Send output for USART4.
					IO	**U4_SCLK —** Serial clock input/output for USART4 in synchronous mode.
PIO2_11	-	-	24 [6]	I; PU	IO	**PIO2_11 —** General-purpose digital input/output pin.
					I	**U4_RXD —** Receiver input for USART4.
PIO2_12	-	-	25 [6]	I; PU	IO	**PIO2_12 —** General-purpose digital input/output pin.
					O	**U4_TXD —** Transmitter output for USART4.
PIO2_13	-	-	26 [6]	I; PU	IO	**PIO2_13 —** General-purpose digital input/output pin.
					I	**U4_CTS —** Clear To Send input for USART4.
PIO2_14	-	-	27 [6]	I; PU	IO	**PIO2_14 —** General-purpose digital input/output pin.
					I	**SCT1_IN2 —** SCTimer1/PWM input 2.
PIO2_15	-	32	49 [6]	I; PU	IO	**PIO2_15 —** General-purpose digital input/output pin.
					I	**SCT1_IN3 —** SCTimer1/PWM input 3.
PIO2_16	-	-	50 [6]	I; PU	IO	**PIO2_16 —** General-purpose digital input/output pin.
					O	**SCT1_OUT0 —** SCTimer1/PWM output 0.
PIO2_17	-	-	51 [6]	I; PU	IO	**PIO2_17 —** General-purpose digital input/output pin.
					O	**SCT1_OUT1 —** SCTimer1/PWM output 1.
PIO2_18	-	33	52 [6]	I; PU	IO	**PIO2_18 —** General-purpose port 2 input/output 18.
					O	**SCT1_OUT2 —** SCTimer1/PWM output 2.
PIO2_19	-	36	57 [6]	I; PU	IO	**PIO2_19 —** General-purpose port 2 input/output 19.
					O	**SCT1_OUT3 —** SCTimer1/PWM output 3.
PIO2_20	-	-	75 [6]	I; PU	IO	**PIO2_20 —** General-purpose port 2 input/output 20.
PIO2_21	-	-	76 [6]	I; PU	IO	**PIO2_21 —** General-purpose port 2 input/output 21.
PIO2_22	-	-	77 [6]	I; PU	IO	**PIO2_22 —** General-purpose port 2 input/output 22.
PIO2_23	-	-	1 [6]	I; PU	IO	**PIO2_23 —** General-purpose port 2 input/output 23.
RSTOUT	-	-	88 [6]	IA	IO	Internal reset status output.
USB_DP	20	26	39 [9]	F	-	USB bidirectional D+ line. Pad includes internal 33 Ω series termination resistor.
USB_DM	19	25	38 [9]	F	-	USB bidirectional D– line. Pad includes internal 33 Ω series termination resistor.
RTCXIN	48	1	5 [2]	-	-	RTC oscillator input. This input should be grounded if the RTC is not used.
RTCXOUT	1	2	6 [2]	-	-	RTC oscillator output.

Table 494. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART3 and USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100	Reset state[1]	Type	Description of pin functions
VREFP	34	47	73	-	-	ADC positive reference voltage. If the ADC is not used, tie VREFP to V_{DD}.
VREFN	35	48	74	-	-	ADC negative voltage reference. If the ADC is not used, tie VREFN to V_{SS}.
V_{DDA}	40	53	84	-	-	Analog voltage supply. V_{DDA}should typically be the same voltages as V_{DD} but should be isolated to minimize noise and error. V_{DDA} should be tied to V_{DD} if the ADC is not used.
V_{DD}	44, 8	58, 10, 34, 59	92, 14, 71, 54, 93	-	-	Supply voltage to the internal regulator and the external rail.
VBAT	47	63	99	-	-	Battery supply. Supplies power to the RTC. If no battery is used, tie VBAT to VDD.
V_{SSA}	41	54	85	-	-	Analog ground. V_{SSA}should typically be the same voltage as V_{SS} but should be isolated to minimize noise and error. V_{SSA} should be tied to V_{SS} if the ADC is not used.
V_{SS}	43, 2, 5	57, 3, 7	91, 7, 11, 53, 70	-	-	Ground.

[1] Pin state at reset for default function: I = Input; AI = Analog Input; O = Output; PU = internal pull-up enabled; IA = inactive, no pull-up/down enabled;
F = floating; If the pins are not used, tie floating pins to ground or power to minimize power consumption.

[2] Special analog pad.

[3] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors, configurable hysteresis, and analog input. When configured as analog input, digital section of the pad is disabled and the pin is not 5 V tolerant; includes digital, programmable filter.

[4] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors, configurable hysteresis, and analog input. When configured as analog input, digital section of the pad is disabled and the pin is not 5 V tolerant; includes digital input glitch filter. WAKEUP pin. The wake-up pin function can be disabled and the pin can be used for other purposes if the RTC is enabled for waking up the part from Deep power-down mode.

[5] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors and configurable hysteresis; includes high-current output driver.

[6] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors and configurable hysteresis.

[7] I^2C-bus pin compliant with the I^2C-bus specification for I^2C standard mode, I^2C Fast-mode, and I^2C Fast-mode Plus. The pin requires an external pull-up to provide output functionality. When power is switched off, this pin is floating and does not disturb the I2C lines. Open-drain configuration applies to all functions on this pin.

[8] 5 V tolerant pad. $\overline{\text{RESET}}$ 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.

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

[10] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors, configurable hysteresis, and analog crystal oscillator connections. When configured for the crystal oscillator input/output, digital section of the pad is disabled and the pin is not 5 V tolerant; includes digital, programmable filter.

35.1.2 LPC11E6x Pin description

Table 495. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100		Reset state[1]	Type	Description of pin functions
$\overline{\text{RESET}}$/PIO0_0	3	4	8	[8]	I; PU	I	**$\overline{\text{RESET}}$** — External reset input with 20 ns glitch filter. A LOW-going pulse as short as 50 ns on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0. This pin also serves as the debug select input. LOW level selects the JTAG boundary scan. HIGH level selects the ARM SWD debug mode. In deep power-down mode, this pin must be pulled HIGH externally. The $\overline{\text{RESET}}$ pin can be left unconnected or be used as a GPIO pin if an external $\overline{\text{RESET}}$ function is not needed and Deep power-down is not used.
						IO	**PIO0_0** — General-purpose digital input/output pin.
PIO0_1	4	5	9	[6]	I; PU	IO	**PIO0_1** — General-purpose digital input/output pin. A LOW level on this pin during reset starts the ISP command handler.
						O	**CLKOUT** — Clockout pin.
						O	**CT32B0_MAT2** — Match output 2 for 32-bit timer 0.
PIO0_2	11	14	19	[6]	I; PU	IO	**PIO0_2** — General-purpose port 0 input/output 2.
						IO	**SSP0_SSEL** — Slave select for SSP0.
						I	**CT16B0_CAP0** — Capture input 0 for 16-bit timer 0.
						-	**R_0** — Reserved.
PIO0_3	14	19	30	[6]	I; PU	IO	**PIO0_3** — General-purpose digital input/output pin.
						-	**R** — Reserved.
							R_1 — Reserved.
PIO0_4	15	20	31	[7]	IA	IO	**PIO0_4** — General-purpose port 0 input/output 4 (open-drain).
						IO	**I2C0_SCL** — I^{2}C-bus clock input/output (open-drain). High-current sink only if I^{2}C Fast-mode Plus is selected in the I/O configuration register.
						-	**R_2** — Reserved.
PIO0_5	16	21	32	[7]	IA	IO	**PIO0_5** — General-purpose port 0 input/output 5 (open-drain).
						IO	**I2C0_SDA** — I^{2}C-bus data input/output (open-drain). High-current sink only if I^{2}C Fast-mode Plus is selected in the I/O configuration register.
						-	**R_3** — Reserved.
PIO0_6	23	29	44	[6]	I; PU	IO	**PIO0_6** — General-purpose port 0 input/output 6.
						-	**R** — Reserved.
						IO	**SSP0_SCK** — Serial clock for SSP0.
						-	**R_4** — Reserved.

Table 495. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100	Reset state[1]	Type	Description of pin functions
PIO0_7	24	30	45 [5]	I; PU	IO	**PIO0_7 —** General-purpose port 0 input/output 7 (high-current output driver).
					I	**U0_CTS —** Clear To Send input for USART.
					-	**R_5 —** Reserved.
					IO	**I2C1_SCL —** I^{2}C-bus clock input/output. This pin is not open-drain.
PIO0_8	26	37	58 [6]	I; PU	IO	**PIO0_8 —** General-purpose port 0 input/output 8.
					IO	**SSP0_MISO —** Master In Slave Out for SSP0.
					O	**CT16B0_MAT0 —** Match output 0 for 16-bit timer 0.
					-	**R_6 —** Reserved.
PIO0_9	27	38	59 [6]	I; PU	IO	**PIO0_9 —** General-purpose port 0 input/output 9.
					IO	**SSP0_MOSI —** Master Out Slave In for SSP0.
					O	**CT16B0_MAT1 —** Match output 1 for 16-bit timer 0.
					-	**R_7 —** Reserved.
SWCLK/PIO0_10	28	39	60 [6]	I; PU	IO	**SWCLK —** Serial Wire Clock. SWCLK is enabled by default on this pin. In boundary scan mode: TCK (Test Clock).
					IO	**PIO0_10 —** General-purpose digital input/output pin.
					IO	**SSP0_SCK —** Serial clock for SSP0.
					O	**CT16B0_MAT2 —** 16-bit timer0 MAT2
TDI/PIO0_11	30	42	64 [3]	I; PU	IO	**TDI —** Test Data In for JTAG interface. In boundary scan mode only.
					IO	**PIO0_11 —** General-purpose digital input/output pin.
					AI	**ADC_9 —** A/D converter, input channel 9.
					O	**CT32B0_MAT3 —** Match output 3 for 32-bit timer 0.
					O	**U1_RTS —** Request To Send output for USART1.
					IO	**U1_SCLK —** Serial clock input/output for USART1 in synchronous mode.
TMS/PIO0_12	31	43	66 [3]	I; PU	IO	**TMS —** Test Mode Select for JTAG interface. In boundary scan mode only.
					IO	**PIO0_12 —** General-purpose digital input/output pin.
					AI	**ADC_8 —** A/D converter, input channel 8.
					I	**CT32B1_CAP0 —** Capture input 0 for 32-bit timer 1.
					I	**U1_CTS —** Clear To Send input for USART1.
TDO/PIO0_13	32	45	68 [3]	I; PU	IO	**TDO —** Test Data Out for JTAG interface. In boundary scan mode only.
					IO	**PIO0_13 —** General-purpose digital input/output pin.
					AI	**ADC_7 —** A/D converter, input channel 7.
					O	**CT32B1_MAT0 —** Match output 0 for 32-bit timer 1.
					I	**U1_RXD —** Receiver input for USART1.

Table 495. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100		Reset state[1]	Type	Description of pin functions
$\overline{\text{TRST}}$/PIO0_14	33	46	69	[3]	I; PU	IO	**$\overline{\text{TRST}}$** — Test Reset for JTAG interface. In boundary scan mode only.
						IO	**PIO0_14** — General-purpose digital input/output pin.
						AI	**ADC_6** — A/D converter, input channel 6.
						O	**CT32B1_MAT1** — Match output 1 for 32-bit timer 1.
						O	**U1_TXD** — Transmitter output for USART1.
SWDIO/PIO0_15	37	50	81	[3]	I; PU	IO	**SWDIO** — Serial Wire Debug I/O. SWDIO is enabled by default on this pin. In boundary scan mode: TMS (Test Mode Select).
						IO	**PIO0_15** — General-purpose digital input/output pin.
						AI	**ADC_3** — A/D converter, input channel 3.
						O	**CT32B1_MAT2** — Match output 2 for 32-bit timer 1.
PIO0_16/WAKEUP	38	51	82	[4]	I; PU	IO	**PIO0_16** — General-purpose digital input/output pin. This pin also serves as the Deep power-down mode wake-up pin with 20 ns glitch filter. Pull this pin HIGH externally before entering 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.
						I	**ADC_2** — A/D converter, input channel 2.
						O	**CT32B1_MAT3** — Match output 3 for 32-bit timer 1.
						-	**R_8** — Reserved.
PIO0_17	42	56	90	[6]	I; PU	IO	**PIO0_17** — General-purpose digital input/output pin.
						O	**U0_$\overline{\text{RTS}}$** — Request To Send output for USART0.
						I	**CT32B0_CAP0** — Capture input 0 for 32-bit timer 0.
						IO	**U0_SCLK** — Serial clock input/output for USART0 in synchronous mode.
PIO0_18	45	60	94	[6]	I; PU	IO	**PIO0_18** — General-purpose digital input/output pin.
						I	**U0_RXD** — Receiver input for USART0. Used in UART ISP mode.
						O	**CT32B0_MAT0** — Match output 0 for 32-bit timer 0.
PIO0_19	46	61	95	[6]	I; PU	IO	**PIO0_19** — General-purpose digital input/output pin.
						O	**U0_TXD** — Transmitter output for USART0. Used in UART ISP mode.
						O	**CT32B0_MAT1** — Match output 1 for 32-bit timer 0.
PIO0_20	10	12	17	[6]	I; PU	IO	**PIO0_20** — General-purpose digital input/output pin.
						I	**CT16B1_CAP0** — Capture input 0 for 16-bit timer 1.
						I	**U2_RXD** — Receiver input for USART2.
PIO0_21	17	22	33	[6]	I; PU	IO	**PIO0_21** — General-purpose digital input/output pin.
						O	**CT16B1_MAT0** — Match output 0 for 16-bit timer 1.
						IO	**SSP1_MOSI** — Master Out Slave In for SSP1.

Table 495. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100		Reset state[1]	Type	Description of pin functions
PIO0_22	29	40	62	[3]	I; PU	IO	**PIO0_22** — General-purpose digital input/output pin.
						AI	**ADC_11** — A/D converter, input channel 11.
						I	**CT16B1_CAP1** — Capture input 1 for 16-bit timer 1.
						IO	**SSP1_MISO** — Master In Slave Out for SSP1.
PIO0_23	39	52	83	[3]	I; PU	IO	**PIO0_23** — General-purpose digital input/output pin.
						AI	**ADC_1** — A/D converter, input channel 1.
						-	**R_9** — Reserved.
						I	**U0_RI** — Ring Indicator input for USART0.
						IO	**SSP1_SSEL** — Slave select for SSP1.
PIO1_0	-	62	97	[6]	I; PU	IO	**PIO1_0** — General-purpose digital input/output pin.
						O	**CT32B1_MAT0** — Match output 0 for 32-bit timer 1.
						-	**R_10** — Reserved.
						O	**U2_TXD** — Transmitter output for USART2.
PIO1_1	-	-	28	[6]	I; PU	IO	**PIO1_1** — General-purpose digital input/output pin.
						O	**CT32B1_MAT1** — Match output 1 for 32-bit timer 1.
						-	**R_11** — Reserved.
						O	**U0_DTR** — Data Terminal Ready output for USART0.
PIO1_2	-	-	55	[6]	I; PU	IO	**PIO1_2** — General-purpose digital input/output pin.
						O	**CT32B1_MAT2** — Match output 2 for 32-bit timer 1.
						-	**R_12** — Reserved.
						I	**U1_RXD** — Receiver input for USART1.
PIO1_3	-	-	72	[3]	I; PU	IO	**PIO1_3** — General-purpose digital input/output pin.
						O	**CT32B1_MAT3** — Match output 3 for 32-bit timer 1.
						-	**R_13** — Reserved.
						IO	**I2C1_SDA** — I^{2}C-bus data input/output (not open-drain).
						AI	**ADC_5** — A/D converter, input channel 5.
PIO1_4	-	-	23	[6]	I; PU	IO	**PIO1_4** — General-purpose digital input/output pin.
						I	**CT32B1_CAP0** — Capture input 0 for 32-bit timer 1.
						-	**R_14** — Reserved.
						I	**U0_DSR** — Data Set Ready input for USART0.
PIO1_5	-	-	47	[6]	I; PU	IO	**PIO1_5** — General-purpose digital input/output pin.
						I	**CT32B1_CAP1** — Capture input 1 for 32-bit timer 1.
						-	**R_15** — Reserved.
						I	**U0_DCD** — Data Carrier Detect input for USART0.
PIO1_6	-	-	98	[6]	I; PU	IO	**PIO1_6** — General-purpose digital input/output pin.
						-	**R_16** — Reserved.
						I	**U2_RXD** — Receiver input for USART2.
						I	**CT32B0_CAP2** — Capture input 2 for 32-bit timer 0.

Table 495. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100	Reset state[1]	Type	Description of pin functions
PIO1_7	-	6	10 [6]	I; PU	IO	**PIO1_7 —** General-purpose digital input/output pin.
					-	**R_17 —** Reserved.
					I	**U2_CTS —** Clear To Send input for USART2.
					I	**CT16B1_CAP0 —** Capture input 0 for 32-bit timer 1.
PIO1_8	-	-	61 [6]	I; PU	IO	**PIO1_8 —** General-purpose digital input/output pin.
					-	**R_18 —** Reserved.
					O	**U1_TXD —** Transmitter output for USART1.
					I	**CT16B0_CAP0 —** Capture input 0 for 16-bit timer 0.
PIO1_9	-	55	86 [3]	I; PU	IO	**PIO1_9 —** General-purpose digital input/output pin.
					I	**U0_CTS —** Clear To Send input for USART0.
					O	**CT16B1_MAT1 —** Match output 1 for 16-bit timer 1.
					AI	**ADC_0 —** A/D converter, input channel 0.
PIO1_10	-	13	18 [6]	I; PU	IO	**PIO1_10 —** General-purpose digital input/output pin.
					O	**U2_RTS —** Request To Send output for USART2.
					IO	**U2_SCLK —** Serial clock input/output for USART2 in synchronous mode.
					O	**CT16B1_MAT0 —** Match output 0 for 16-bit timer 1.
PIO1_11	-	-	65 [6]	I; PU	IO	**PIO1_11 —** General-purpose digital input/output pin.
					IO	**I2C1_SCL —** I^{2}C1-bus clock input/output (not open-drain).
					O	**CT16B0_MAT2 —** Match output 2 for 16-bit timer 0.
					I	**U0_RI —** Ring Indicator input for USART0.
PIO1_12	-	-	89 [6]	I; PU	IO	**PIO1_12 —** General-purpose digital input/output pin.
					IO	**SSP0_MOSI —** Master Out Slave In for SSP0.
					O	**CT16B0_MAT1 —** Match output 1 for 16-bit timer 0.
					-	**R_21 —** Reserved.
PIO1_13	36	49	78 [6]	I; PU	IO	**PIO1_13 —** General-purpose digital input/output pin.
					I	**U1_CTS —** Clear To Send input for USART1.
					O	**SCT0_OUT3 —** SCTimer0/PWM output 3.
					-	**R_22 —** Reserved.
PIO1_14	-	-	79 [6]	I; PU	IO	**PIO1_14 —** General-purpose digital input/output pin.
					IO	**I2C1_SDA —** I^{2}C1-bus data input/output (not open-drain).
					O	**CT32B1_MAT2 —** Match output 2 for 32-bit timer 1.
					-	**R_23 —** Reserved.
PIO1_15	-	-	87 [6]	I; PU	IO	**PIO1_15 —** General-purpose digital input/output pin.
					IO	**SSP0_SSEL —** Slave select for SSP0.
					O	**CT32B1_MAT3 —** Match output 3 for 32-bit timer 1.
					-	**R_24 —** Reserved.

Table 495. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100	Reset state[1]	Type	Description of pin functions
PIO1_16	-	-	96 [6]	I; PU	IO	**PIO1_16 —** General-purpose digital input/output pin.
					IO	**SSP0_MISO —** Master In Slave Out for SSP0.
					O	**CT16B0_MAT0 —** Match output 0 for 16-bit timer 0.
					-	**R_25 —** Reserved.
PIO1_17	-	-	34 [6]	I; PU	IO	**PIO1_17 —** General-purpose digital input/output pin.
					I	**CT16B0_CAP2 —** Capture input 2 for 16-bit timer 0.
					I	**U0_RXD —** Receiver input for USART0.
					-	**R_26 —** Reserved.
PIO1_18	-	-	43 [6]	I; PU	IO	**PIO1_18 —** General-purpose digital input/output pin.
					I	**CT16B1_CAP1 —** Capture input 1 for 16-bit timer 1.
					O	**U0_TXD —** Transmitter output for USART0.
					-	**R_27 —** Reserved.
PIO1_19	-	64	4 [6]	I; PU	IO	**PIO1_19 —** General-purpose digital input/output pin.
					I	**U2_CTS —** Clear To Send input for USART2.
					O	**SCT0_OUT0 —** SCTimer0/PWM output 0.
					-	**R_28 —** Reserved.
PIO1_20	13	18	29 [6]	I; PU	IO	**PIO1_20 —** General-purpose digital input/output pin.
					I	**U0_DSR —** Data Set Ready input for USART0.
					IO	**SSP1_SCK —** Serial clock for SSP1.
					O	**CT16B0_MAT0 —** Match output 0 for 16-bit timer 0.
PIO1_21	25	35	56 [6]	I; PU	IO	**PIO1_21 —** General-purpose digital input/output pin.
					I	**U0_DCD —** Data Carrier Detect input for USART0.
					IO	**SSP1_MISO —** Master In Slave Out for SSP1.
					I	**CT16B0_CAP2 —** Capture input 2 for 16-bit timer 0.
PIO1_22	-	-	80 [3]	I; PU	IO	**PIO1_22 —** General-purpose digital input/output pin.
					IO	**SSP1_MOSI —** Master Out Slave In for SSP1.
					I	**CT32B1_CAP1 —** Capture input 1 for 32-bit timer 1.
					AI	**ADC_4 —** A/D converter, input channel 4.
					-	**R_29 —** Reserved.
PIO1_23	18	23	35 [6]	I; PU	IO	**PIO1_23 —** General-purpose digital input/output pin.
					O	**CT16B1_MAT1 —** Match output 1 for 16-bit timer 1.
					IO	**SSP1_SSEL —** Slave select for SSP1.
					O	**U2_TXD —** Transmitter output for USART2.
PIO1_24	22	28	42 [6]	I; PU	IO	**PIO1_24 —** General-purpose digital input/output pin.
					O	**CT32B0_MAT0 —** Match output 0 for 32-bit timer 0.
					IO	**I2C1_SDA —** I^{2}C-bus data input/output (not open-drain).

Table 495. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100	Reset state[1]	Type	Description of pin functions
PIO1_25	-	-	100 [6]	I; PU	IO	**PIO1_25 —** General-purpose digital input/output pin.
					O	**U2_$\overline{\text{RTS}}$ —** Request To Send output for USART2.
					IO	**U2_SCLK —** Serial clock input/output for USART2 in synchronous mode.
					I	**SCT0_IN0 —** SCTimer0/PWM input 0.
					-	**R_30 —** Reserved.
PIO1_26	-	15	20 [6]	I; PU	IO	**PIO1_26 —** General-purpose digital input/output pin.
					O	**CT32B0_MAT2 —** Match output 2 for 32-bit timer 0.
					I	**U0_RXD —** Receiver input for USART0.
					-	**R_19 —** Reserved.
PIO1_27	-	17	22 [6]	I; PU	IO	**PIO1_27 —** General-purpose digital input/output pin.
					O	**CT32B0_MAT3 —** Match output 3 for 32-bit timer 0.
					O	**U0_TXD —** Transmitter output for USART0.
					-	**R_20 —** Reserved.
					IO	**SSP1_SCK —** Serial clock for SSP1.
PIO1_28	-	31	46 [6]	I; PU	IO	**PIO1_28 —** General-purpose digital input/output pin.
					I	**CT32B0_CAP0 —** Capture input 0 for 32-bit timer 0.
					IO	**U0_SCLK —** Serial clock input/output for USART in synchronous mode.
					O	**U0_$\overline{\text{RTS}}$ —** Request To Send output for USART0.
PIO1_29	-	41	63 [3]	I; PU	IO	**PIO1_29 —** General-purpose digital input/output pin.
					IO	**SSP0_SCK —** Serial clock for SSP0.
					I	**CT32B0_CAP2 —** Capture input 2 for 32-bit timer 0.
					O	**U0_$\overline{\text{DTR}}$ —** Data Terminal Ready output for USART0.
					AI	**ADC_10 —** A/D converter, input channel 10.
PIO1_30	-	44	67 [6]	I; PU	IO	**PIO1_30 —** General-purpose digital input/output pin.
					IO	**I2C1_SCL —** I^{2}C1-bus clock input/output (not open-drain).
					I	**SCT0_IN3 —** SCTimer0/PWM input 3.
					-	**R_31 —** Reserved.
PIO1_31	-	-	48 [5]	I; PU	IO	**PIO1_31 —** General-purpose digital input/output pin (high-current output driver).
PIO2_0	6	8	12 [10]	I; PU	IO	**PIO2_0 —** General-purpose digital input/output pin.
					AI	**XTALIN —** Input to the oscillator circuit and internal clock generator circuits. Input voltage must not exceed 1.8 V.
PIO2_1	7	9	13 [10]	I; PU	IO	**PIO2_1 —** General-purpose digital input/output pin.
					AO	**XTALOUT —** Output from the oscillator amplifier.

Table 495. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100		Reset state[1]	Type	Description of pin functions
PIO2_2	12	16	21	[6]	I; PU	IO	**PIO2_2 —** General-purpose digital input/output pin.
						O	**U3_$\overline{\text{RTS}}$ —** Request To Send output for USART3.
						IO	**U3_SCLK —** Serial clock input/output for USART3 in synchronous mode.
						O	**SCT0_OUT1 —** SCTimer0/PWM output 1.
PIO2_3	19	24	36	[6]	I; PU	IO	**PIO2_3 —** General-purpose digital input/output pin.
						I	**U3_RXD —** Receiver input for USART3.
						O	**CT32B0_MAT1 —** Match output 1 for 32-bit timer 0.
PIO2_4	21	27	41	[6]	I; PU	IO	**PIO2_4 —** General-purpose digital input/output pin.
						O	**U3_TXD —** Transmitter output for USART3.
						O	**CT32B0_MAT2 —** Match output 2 for 32-bit timer 0.
PIO2_5	9	11	15	[6]	I; PU	IO	**PIO2_5 —** General-purpose digital input/output pin.
						I	**U3_$\overline{\text{CTS}}$ —** Clear To Send input for USART3.
						I	**SCT0_IN1 —** SCTimer0/PWM input 1.
PIO2_6	-	25	37	[6]	I; PU	IO	**PIO2_6 —** General-purpose digital input/output pin.
						O	**U1_$\overline{\text{RTS}}$ —** Request To Send output for USART1.
						IO	**U1_SCLK —** Serial clock input/output for USART1 in synchronous mode.
						I	**SCT0_IN2 —** SCTimer0/PWM input 2.
PIO2_7	20	26	40	[6]	I; PU	IO	**PIO2_7 —** General-purpose digital input/output pin.
						IO	**SSP0_SCK —** Serial clock for SSP0.
						O	**SCT0_OUT2 —** SCTimer0/PWM output 2.
PIO2_8	-	-	2	[6]	I; PU	IO	**PIO2_8 —** General-purpose digital input/output pin.
						I	**SCT1_IN0 —** SCTimer1/PWM input 0.
PIO2_9	-	-	3	[6]	I; PU	IO	**PIO2_9 —** General-purpose digital input/output pin.
						I	**SCT1_IN1 —** SCTimer1/PWM_IN1
PIO2_10	-	-	16	[6]	I; PU	IO	**PIO2_10 —** General-purpose digital input/output pin.
						O	**U4_$\overline{\text{RTS}}$ —** Request To Send output for USART4.
						IO	**U4_SCLK —** Serial clock input/output for USART4 in synchronous mode.
PIO2_11	-	-	24	[6]	I; PU	IO	**PIO2_11 —** General-purpose digital input/output pin.
						I	**U4_RXD —** Receiver input for USART4.
PIO2_12	-	-	25	[6]	I; PU	IO	**PIO2_12 —** General-purpose digital input/output pin.
						O	**U4_TXD —** Transmitter output for USART4.
PIO2_13	-	-	26	[6]	I; PU	IO	**PIO2_13 —** General-purpose digital input/output pin.
						I	**U4_$\overline{\text{CTS}}$ —** Clear To Send input for USART4.
PIO2_14	-	-	27	[6]	I; PU	IO	**PIO2_14 —** General-purpose digital input/output pin.
						I	**SCT1_IN2 —** SCTimer1/PWM input 2.

Table 495. Pin description

Pin functions are selected through the IOCON registers. See Table 2 for availability of USART4 pin functions.

Symbol	LQFP48	LQFP64	LQFP100		Reset state[1]	Type	Description of pin functions
PIO2_15	-	32	49	[6]	I; PU	IO	**PIO2_15 —** General-purpose digital input/output pin.
						I	**SCT1_IN3 —** SCTimer1/PWM input 3.
PIO2_16	-	-	50	[6]	I; PU	IO	**PIO2_16 —** General-purpose digital input/output pin.
						O	**SCT1_OUT0 —** SCTimer1/PWM output 0.
PIO2_17	-	-	51	[6]	I; PU	IO	**PIO2_17 —** General-purpose digital input/output pin.
						O	**SCT1_OUT1 —** SCTimer1/PWM output 1.
PIO2_18	-	33	52	[6]	I; PU	IO	**PIO2_18 —** General-purpose port 2 input/output 18.
						O	**SCT1_OUT2 —** SCTimer1/PWM output 2.
PIO2_19	-	36	57	[6]	I; PU	IO	**PIO2_19 —** General-purpose port 2 input/output 19.
						O	**SCT1_OUT3 —** SCTimer1/PWM output 3.
PIO2_20	-	-	75	[6]	I; PU	IO	**PIO2_20 —** General-purpose port 2 input/output 20.
PIO2_21	-	-	76	[6]	I; PU	IO	**PIO2_21 —** General-purpose port 2 input/output 21.
PIO2_22	-	-	77	[6]	I; PU	IO	**PIO2_22 —** General-purpose port 2 input/output 22.
PIO2_23	-	-	1	[6]	I; PU	IO	**PIO2_23 —** General-purpose port 2 input/output 23.
$\overline{\text{RSTOUT}}$	-	-	88	[6]	IA	IO	Internal reset status output.
RTCXIN	48	1	5	[2]	-	-	RTC oscillator input. This input should be grounded if the RTC is not used.
RTCXOUT	1	2	6	[2]	-	-	RTC oscillator output.
VREFP	34	47	73		-	-	ADC positive reference voltage. If the ADC is not used, tie VREFP to V_{DD}.
VREFN	35	48	74		-	-	ADC negative voltage reference. If the ADC is not used, tie VREFN to V_{SS}.
V_{DDA}	40	53	84		-	-	Analog voltage supply. V_{DDA}should typically be the same voltages as V_{DD} but should be isolated to minimize noise and error. V_{DDA} should be tied to V_{DD} if the ADC is not used.
V_{DD}	44, 8	58, 10, 34, 59	92, 14, 71, 54, 93		-	-	Supply voltage to the internal regulator and the external rail.
VBAT	47	63	99		-	-	Battery supply. Supplies power to the RTC. If no battery is used, tie VBAT to VDD.
V_{SSA}	41	54	85		-	-	Analog ground. V_{SSA}should typically be the same voltage as V_{SS} but should be isolated to minimize noise and error. V_{SSA} should be tied to V_{SS} if the ADC is not used.
V_{SS}	43, 2, 5	57, 3, 7	91, 7, 11, 53, 70		-	-	Ground.
n.c.	-	-	39				Not connected.
n.c.	-	-	38				Not connected.

[1] Pin state at reset for default function: I = Input; O = Output; AI = Analog Input; PU = internal pull-up enabled; IA = inactive, no pull-up/down enabled;
F = floating; If the pins are not used, tie floating pins to ground or power to minimize power consumption.

[2] Special analog pad.

[3] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors, configurable hysteresis, and analog input. When configured as analog input, digital section of the pad is disabled and the pin is not 5 V tolerant; includes digital, programmable filter.

[4] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors, configurable hysteresis, and analog input. When configured as analog input, digital section of the pad is disabled and the pin is not 5 V tolerant; includes digital input glitch filter. WAKEUP pin. The wake-up pin function can be disabled and the pin can be used for other purposes if the RTC is enabled for waking up the part from Deep power-down mode.

[5] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors and configurable hysteresis; includes high-current output driver.

[6] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors and configurable hysteresis.

[7] I^2C-bus pin compliant with the I^2C-bus specification for I^2C standard mode, I^2C Fast-mode, and I^2C Fast-mode Plus. The pin requires an external pull-up to provide output functionality. When power is switched off, this pin is floating and does not disturb the I2C lines. Open-drain configuration applies to all functions on this pin.

[8] 5 V tolerant pad. $\overline{\text{RESET}}$ 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.

[9] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors, configurable hysteresis, and analog crystal oscillator connections. When configured for the crystal oscillator input/output, digital section of the pad is disabled and the pin is not 5 V tolerant; includes digital, programmable filter.

UM10732

Chapter 36: LPC11U6x/E6x Serial Wire Debug (SWD)

Rev. 1.8 — 19 December 2016 User manual

36.1 How to read this chapter

The debug functionality is identical for all parts.

36.2 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 watch points that can also be used as triggers.
- Supports JTAG boundary scan.
- Micro Trace Buffer (MTB) supported.

36.3 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 watch points to support debug and trace.

Support for boundary scan and Micro Trace Buffer is available.

The debugger can access all memories, registers and peripherals. Inside the CortexM0+, there is an embedded ROM table for identifying the debug CoreSight components which are located inside the core.

36.4 Pin description

The SWD functions are enabled by default.

Table 496. SWD pin description

Function	Type	Pin	Description
SWCLK	I/O	SWCLK/PIO0_10/ SSP0_SCK/ CT16B0_MAT2	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.
SWDIO	I/O	SWDIO/PIO0_15/ADC_3/ CT32B1_MAT2	**Serial wire debug data input/output.** The SWDIO pin is used by an external debug tool to communicate with and control the part. This pin is pulled up internally.

Table 497. JTAG boundary scan pin description

Function	Pin name	Type	Description
TCK	SWCLK/PIO0_10/SSP0_SCK/ CT16B0_MAT2	I	**JTAG Test Clock.** This pin is the clock for JTAG boundary scan when the RESET pin is LOW.
TMS	TMS/PIO0_12/ADC_8/CT32B1_CAP0/ U1_nCTS	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	TDI/PIO0_11/ADC_9/CT32B0_MAT3/ U1_nRTS/U1_SCLK	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	TDO/PIO0_13/ADC_7/CT32B1_MAT0/ U1_RXD	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	nTRST/PIO0_14/ADC_6/ CT32B1_MAT1/U1_TXD	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.

36.5 Functional description

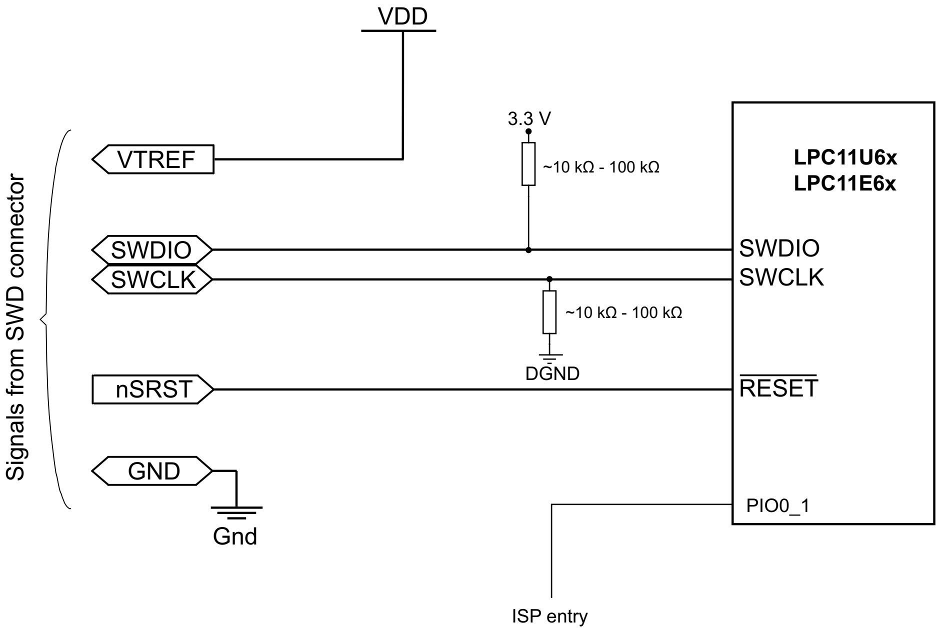
36.5.1 Debug limitations

It is recommended not to use the debug mode during Deep-sleep or Power-down mode.

During a debugging session, the System Tick Timer is automatically stopped whenever the CPU is stopped. Other peripherals are not affected.

36.5.2 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 98. Connecting the SWD pins to a standard SWD connector

36.5.3 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.

36.5.4 Micro Trace Buffer (MTB)

The MTB (mciro-trace-buffer) is integrated outside the Cortex M0+ core. MTB is not only used for debug control between the Cortex M0+ execution interface and the main SRAM but also for debug control between the system AHB bus interface and the main SRAM. It has the following features:

- Provision of program flow tracing for the Cortex M0+ processor.
- MTB SRAM can be used for both trace and general purpose storage by the processor.
- The position and size of the trace buffer in SRAM is configurable by software.
- External hardware can control trace start/stop.
- CoreSight compliant .

When MTB is enabled, the MTB records change in the program flow, reported by the Cortex M0+ processor over the execution trace interface. This information is stored as trace packets in the SRAM. An off-chip debugger can extract the trace information using the DAP to read the trace information from the SRAM over the AHB-lite interface. The debugger can then reconstruct the program flow from this information.

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 4.4.32) 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.

To support the automatic SWD detection of the MTB, one system ROM table is available. This system ROM table includes two entries, one point to the embedded ROM table inside CortexM0+, another point to MTB SFR base address.

The system ROM table is located at address 0x5000 8000.

37.1 Abbreviations

Table 498. Abbreviations

Acronym	Description
AHB	Advanced High-performance Bus
APB	Advanced Peripheral Bus
BOD	BrownOut Detection
GPIO	General-Purpose Input/Output
PLL	Phase-Locked Loop
SPI	Serial Peripheral Interface
SMBus	System Management Bus
UART	Universal Asynchronous Receiver/Transmitter

37.2 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

[4] LPC11U6x data sheet: http://www.nxp.com/documents/data_sheet/LPC11U6X.pdf

[5] LPC11U6x Errata sheet: http://www.nxp.com/documents/errata_sheet/ES_LPC11U6X.pdf

[6] LPC11E6x data sheet: http://www.nxp.com/documents/data_sheet/LPC11E6X.pdf

[7] LPC11E6x Errata sheet: http://www.nxp.com/documents/errata_sheet/ES_LPC11E6X.pdf

37.3 Legal information

37.3.1 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.

37.3.2 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.

37.3.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

I^{2}C-bus — logo is a trademark of NXP B.V.

37.4 Tables

37.5 Figures

37.6 Contents

Chapter 1: LPC11U6x/E6x Introductory information

Chapter 2: LPC11U6x/E6x Memory map

Chapter 3: LPC11U6x/E6x Nested Vectored Interrupt Controller (NVIC)

Chapter 4: LPC11U6x/E6x System configuration (SYSCON)

Chapter 5: LPC11U6x/E6x Power Management Unit (PMU)

Chapter 6: LPC11U6x/E6x I/O control (IOCON)

Chapter 7: LPC11U6x/E6x General-Purpose I/O (GPIO)

Chapter 8: LPC11U6x/E6x Grouped GPIO input interrupt (GINT0/1)

Chapter 9: LPC11U6x/E6x Pin interrupt and pattern match engine (PINT)

Chapter 10: LPC11U6x/E6x DMA controller

Chapter 11: LPC11U6x/E6x USART0

Chapter 12: LPC11U6x/E6x USART1/2/3/4

Chapter 13: LPC11U6x/E6x I2C0/1-bus interfaces

Chapter 14: LPC11U6x/E6x SSP0/1

Chapter 15: LPC11U6x USB2.0 full-speed device controller

Chapter 16: LPC11U6x/E6x 12-bit Analog-to-Digital Converter (ADC)

Chapter 17: LPC11U6x/Ex Temperature sensor

Chapter 18: LPC11U6x/E6x SCtimer/PWM (SCT0/1)

Chapter 19: LPC11U6x/E6x 16-bit counter/timers CT16B0/1

Chapter 20: LPC11U6x/E6x 32-bit counter/timers CT32B0/1

Chapter 21: LPC11U6x/E6x Real-Time Clock (RTC)

Chapter 22: LPC11U6x/E6x Windowed Watchdog Timer (WWDT)

Chapter 23: LPC11U6x/E6x Cyclic Redundancy Check (CRC) engine

Chapter 24: LPC11U6x/E6x System tick timer (SysTick)

Chapter 25: LPC11U6x/E6x Flash controller

Chapter 26: LPC11U6x/E6x Boot ROM

Chapter 27: LPC11U6x/E6x Flash/EEPROM ISP/IAP programming

Chapter 28: LPC11U6x/E6x Power profiles/Power control API

Chapter 29: LPC11U6x/E6x Integer division routines

Chapter 30: LPC11U6x/E6x I2C-bus ROM API

Chapter 31: LPC11U6x/E6x USART ROM API (USART0)

Chapter 32: LPC11U6x/E6x USART ROM API (USART1/2/3/4)

Chapter 33: LPC11U6x/E6x DMA ROM API

Chapter 34: LPC11U6x USB ROM API

Chapter 35: LPC11U6x/Ex Pin description

Chapter 36: LPC11U6x/E6x Serial Wire Debug (SWD)

Chapter 37: Supplementary information

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section ‘Legal information’.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 19 December 2016
Document identifier: UM10732