UM11126

LPC55S6x/LPC55S2x/LPC552x User manual

Rev. 2.7 — 3 October 2024 User manual

Document information

Info	Content
Keywords	LPC55S6x/LPC55S2x/LPC552x, ARM Cortex-M33 core, 32-bit microcontroller, SCTimer/PWM, PLU, TrustZone, PowerQuad, CASPER, USB Host, USB device, I2C, AES, PUF, SHA, CRC, RNG, 16-bit ADC
Abstract	LPC55S6x/LPC55S2x/LPC552x User Manual

Revision history

Document ID	Release Date	Description
UM11126 v. 2.7	3 October 2024	• Added VFBGA59 package details in Section 1.2 “Features”. • Added VFBGA59 package details in Table 1 “Ordering information” and Table 2 “Ordering options”. • Added footnote in Table 56 • Added footnote in Table 57 • Added footnote in Table 311
UM11126 v. 2.6	19 December 2023	• Added a note in Section 48.9 “DICE”. • Updated Section 8.11.1.1 “Device descriptor”. • Updated CMD[CTYPE] register definition field in Table 744 “ADC command low buffer registers (CMDL[1:15], offsets 0x100 to 0x170)”.

Contact information

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

Revision history *…continued*

Document ID	Release Date	Description
UM11126 v. 2.5	16 June 2023	• In Chapter 1 “LPC55S6x/LPC55S2x/LPC552x Introductory Information”, replaced LPC55S6x/LPC55S2x/LPC552x to LPC55S6x/LPC55S2x also offers support for HASH, AES, RSA, and so on. • In Section 4.5.49 “Fractional rate divider for each Flexcomm Interface frequency”, Section 32.4 “Basic configuration”, Section 33.4 “Basic configuration”, Section 34.3 “Basic configuration”, Section 35.3 “Basic configuration”, and Section 37.3 “Basic configuration”updated the FRG maximum allowed output frequency. • In Table 153 of Section 4.5.88 “Device ID register”, updated device ids for LPC55S26, LPC55S28, LPC5526, and LPC5528. • In Section 5.3 “Block diagram”, updated “Remark”. • Section 6.4.3 “SPI flash recovery”, updated to read “If SEC_BOOT_EN is non-zero, then the image can be booted into internal SRAM in a non-reserved region”. • In Chapter 7 “LPC55S6x/LPC55S2x/LPC552x Secure Boot ROM”, updated LPC55S6x/LPC55S2x series. • In Table 245 “Properties used by Get/SetProperty commands, sorted by values”, removed SystemDeviceid. • In Section 10.2.1 “Customer Manufacturing Programmable Area (CMPA)”, added default pin for boot failure indication using GPIO port pin. • In Chapter 10 “LPC55S6x/LPC55S2x/LPC552x Protected Flash Region”, made editorial updates. • In “Transitions” column of Table 273 “Lifecycle state descriptions” and in Figure 47 “Customer Development Lifecycle State”, replaced “RETURN_EN” with “ENABLE_FA_MODE”. • In Table 292 “Power domain supply”, added table note 1 “The GPIO logic level does not remain static in power-down mode. All GPIO pin state will be logic '0' in power-down mode” for Analog components and updated power domain system for Digital components. • In Section 13.2.3 “Power modes” and Section 13.3.4 “Deep-sleep mode” added GPIO logic in deep-sleep mode. • Updated Section 13.3.5 “Power-down mode”. • In Table 520 “Suggested SCT input pin settings” and Table 521 “Suggested SCT output pin settings”, updated IOCON 9, 10, and 11 bits. • In Table 644 “Suggested USART pin settings”, updated pin details for 11 to 15 IOCON bits. • In Table 668 “Suggested SPI pin settings”, updated pin details for 9 to 15 IOCON bits. • In Section 39.3 “Basic configuration”, added Remarks. • In Table 741 “ADC FIFO control registers (FCTRL[0:1], offsets 0xE0 to 0xE4)”, FWMARK description updated. • FRO1M removed from Section 39.7.4 “Clock operation”. • Remarks added in Section 39.7.6 “Temperature sensor”. • Crystal oscillator value corrected to read 16 MHz instead of 12 MHz for operating frequency, and 25 MHz instead of 24 MHz for clock frequency in Chapter 1 “LPC55S6x/LPC55S2x/LPC552x Introductory Information”, Section 11.2 “Features”, Table 275 “Register overview: ANACTRL (base address = 0x50013000)”, Section 11.5.8 “32 MHz crystal oscillator status register”, and Section 11.5.11 “High Speed Crystal Oscillator (16 MHz - 32 MHz) Voltage Source Supply Control register (LDO_XO32 M)”. • Updated Table 644 “Suggested USART pin settings” and Table 668 “Suggested SPI pin settings”. • SYSAHBCLKCTRL2 corrected to “To save power, clear USB1_HOST in **AHBCLKCTRL2**” in Chapter 44 “LPC55S6x/LPC55S2x/LPC552x USB1 High-Speed Device Controller”. • P11 corrected to PLL (480MHz) in Figure 166 “USB 2.0 PHY block diagram”. • Updated Table 852 “USB PHY Transmitter Control Register (TX, offset 0x10)”. • Added “PLL lock bit errata” remark in Section 4.5.69.1.2 “PLL0 status register” and Section 4.6.6.2.1 “Lock detector”.

Revision history *…continued*

Document ID	Release Date	Description
UM11126 v. 2.4	7 October 2021	• CPU0 and CPU1 usage - low power modes details added in Section 13.2 “General description”. • In Chapter 4 “LPC55S6x/LPC55S2x/LPC552x SYSCON” added Section 4.6.6.3.4 “Fractional Divide operation”. • ENVADJ value updated in Table 856 “USB PHY Receiver Control Register (RX, offset 0x20)”, Table 857 “USB PHY Receiver Control Register (RX_SET, offset 0x24)”, and Table 858 “USB PHY Receiver Control Register (RX_CLR, offset 0x28)”. • Updated LOCK_REG name and description for bits 21:20, 19:18, and 17:16 in Table 183 “PRINCE configuration”. • In Section 4.5.69.1.5 “Spread spectrum control with the System PLL” added Table 127 “Modulator input (spread spectrum enabled): md => F_{clkcco}= (md[32:25]dec + md[24:0]dec*2^{-25})*F_{ref}” and Table 128 “Modulator input (spread spectrum disabled, only fractional => mc=0, mr=0): md => F_{clkcco} = (md[32:25]dec + md[24:0]dec*2^{-25}) * F_{ref}”. • Bit 21 Forced FS value and description updated in Table 834 “USB1 device command/status register (DEVCMDSTAT, offset = 0x000)”. • ADC signal changed from ADC0_0 - ADC0_63 to ADC0_0 - ADC0_N in Table 723 “ADC signal descriptions”, and added Table note ”N is dependent on SoC configuration. Refer Table 725 “ADC Inputs Selection & ADC programming” for details.” • In Table 745 “ADC command high buffer registers (CMDH[1:15], offsets 0x104 to 0x174))” STS field updated to minimum sampling time of ADCK cycles 3.5 seconds. • In Chapter 10 “LPC55S6x/LPC55S2x/LPC552x Protected Flash Region” added Section 10.4 “Firewall PFR pages”. • In Table 4 “Memory map overview” updated AHB port 0 function description to last 16 pages (10 KB) instead of last 17 pages (10 KB). • Logic levels of the pins updated in Section 13.3.3 “Sleep-mode”, Section 13.3.4 “Deep-sleep mode”, and Section 13.3.5 “Power-down mode”. • USB1_CHRGx register removed from Table 847 “Register overview: crr_d_ip_hs_usb2phy_gf40nvrf (base address = 0x50038000)” as it is not supported by LPC55S6x device. • Footnote added for LOCK_ALL reset value in Table 145 “Debug Lock Enable (DEBUG_LOCK_EN, offset = 0xFA0)”. • Updated description of DISABLE_VIOLATION_ABORT in Table 965 “Secure control register (MISC_CTRL_REG, offset = 0xFFC)”. • EP buffer address width changed to 16 bits in Figure 154 “Endpoint command/status list (see Table 778)”. • Updated Device_id0 register values in Table 153 “Device ID0 register (DEVICE_ID0, offset = 0xFF8)”. • Added Section 11.5.12 “AUX_BIAS” and AUX_BIAS module description in Table 275 “Register overview: ANACTRL (base address = 0x50013000)”. • In Chapter 45 “LPC55S6x/LPC55S2x/LPC552x USB1 High-Speed PHY” removed registers USB1_CHRG_DETECT, USB1_CHRG_DETECT_SET, USB1_CHRG_DETECT_TOG, and USB1_CHRG_DET_STAT. • In Chapter 4 “LPC55S6x/LPC55S2x/LPC552x SYSCON” added Section 4.5.70 “Functional retention control register (FUNCRETENTIONCTRL)” and updated Table 38 “Register overview: SYSCON (base address = 0x50000000)”. • In Chapter 11 “LPC55S6x/LPC55S2x/LPC552x Analog control” added Section 11.5.1 “Various Analog blocks configuration (like FRO 192MHz trimmings source …) (ANALOG_CTRL_CFG)”, Section 11.5.6 “General Purpose ADC VBAT Divider branch control”, Section 11.5.11 “High Speed Crystal Oscillator (16 MHz - 32 MHz) Voltage Source Supply Control register (LDO_XO32 M)”, Section 11.5.13 “USB High Speed Phy Trim values”, and updated Section 11.5.7 “32 MHz crystal oscillator control register”. Also, updated Table 275 “Register overview: ANACTRL (base address = 0x50013000)”.

Revision history *…continued*

Document ID	Release Date	Description
		• ISP PNG ping packet Figure 43 “Host reads ping packet from target via SPI” updated. • Description of DM-AP commands 0x02 and 0x03 updated in Section 51.5.7.1.1 “DM-AP commands”. • PFR PUF key code updated in Table 179 “PUF key code storage area structure” and Table 180 “PUF key code format”. • Included CFPA memory bank details in Section 10.2 “General description” and added PFR memory details in Table 272 “PFR memory table”. • In Chapter 10 “LPC55S6x/LPC55S2x/LPC552x Protected Flash Region” added Figure 47 “Customer Development Lifecycle State”. • In Chapter 10 “LPC55S6x/LPC55S2x/LPC552x Protected Flash Region” added Section 10.4 “Firewall PFR pages”. • Added 19:16 bits information in Table 332 “Parameter wakeup_io_ctrl”. • Added ADC channel 12, 13, and 26 connection and description in Table 725 “ADC Inputs Selection & ADC programming” and Table 727 “ADC Specific channels”. • Added information in Section 39.7.6 “Temperature sensor” that to use temperature sensor maximum ADC input clock frequency is 6 MHz. • Remark added in USB HS ISP mode in Table 175 “ISP download mode based on DEFAULT_ISP_MODE bits (6:4, word 0 in CMPA)”. • SOCU_PIN [n] and SOCU_DFLT [n] details updated in Table 1081 “Access restriction levels”. • Wakeup pin feature (power-down mode) details added in Table 8 “Connection of interrupt sources to the NVIC”, Section 13.4 “Register description”, Section 13.4.10 “Wake-up I/O Control register”, and Table 323 “Parameter wakeup_interrupts”. • Updated PFR address in Figure 13 “Protected Flash Region”. • Flexcomm 9 removed in Section 1.2 “Features” as it is not used as ISP interface. • Added Power configuration register 0, bit 19 PDEN_AUXBIAS provides power control to BIAS_VREF_1V in Table 311 “Power configuration register 0 (PDRUNCFG0, offset = 0xB8) (Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset)”. • Added BIAS_VREF_IV for AUXBIAS in Table 294 “Peripheral reduced power modes”. • Offset value of NVIC register updated in Table 9 “Register overview: NVIC (base address = 0xe000e100)”. • Divide by 2 remark removed from Section 4.6.6.3.1 “Normal modes”, Figure 8 “PLL block diagram showing typical operation”, and Figure 9 “PLL0 block diagram showing spread spectrum and fractional divide operation”. • Updated benchmark values for Powerquad functions in Table 1049 “Convolution/Correlation/FIR functions”, Table 1050 “Matrix Engine”, and Table 1051 “Register overview (base address 0x400A 6000)”. • SDIOCLKCTRL updated to write only register in Table 38 “Register overview: SYSCON (base address = 0x50000000)”. • Secure boot address updated to 0x9E41C in Section 51.7.1.1 “Protocol Version (DCFG_VER)”, CMPA address updated to 0x9E450 – 0x9E46C in Section 51.7.1.2 “Root of Trust Identifier (DCFG_ROTID)”, CC_SOCU_PIN word PFR address updated to 0x9E410 in Section 51.7.1.3 “Enforce UUID checking (DCFG_UUID)”, and updated CMPA.VENDOR_USAGE PFR address in Section 51.7.1.5 “DCFG_VENDOR_USAGE”. • Added description of CMPA and CFPA in Table ”41.9.7 Glossary”. • Register names CMDL, CMDH and TCTRL updated in Section 39.7.6 “Temperature sensor”. • Added equation for PUDLY to calculate tADCSTUP in Section 39.7.3 “Power control”. • Side Polarity details added in Table 726 “Differential Pairs”. • Added a note in Chapter 43 “LPC55S6x/LPC55S2x/LPC552x USB1 High-Speed Host Controller” MULT restriction in Table 829 “PTD bit definition”.

Revision history *…continued*

Document ID	Release Date	Description
		• In Chapter 13 “LPC55S6x/LPC55S2x/LPC552x Power Management” added Section 13.4.1 “Power Management Controller FSM (Finite State Machines) status (STATUS)”, Section 13.4.3 “DCDC (first) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (DCDC0)”, Section 13.4.4 “DCDC (second) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (DCDC1)”, Section 13.4.5 “Power Management Unit (PMU) and Always-On domains LDO control [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (LDOPMU)”, Section 13.4.7 “Analog References fast wake-up Control register [Reset by: PoR]”, Section 13.4.8 “32 KHz Crystal oscillator (XTAL) control register [Reset by: PoR, Brown Out Detectors Reset]”, Section 13.4.10 “Wake-up I/O Control register”, Section 13.4.14 “Dummy Control bus to PMU [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset]”, and updated Section 13.4.2 “Reset control register”. Also, updated Section 13.4 “Register description”. • Note extended in Section 14.4.4.2 “Param1: sram_retention_ctrl” to read if the user application uses power-down mode then it is recommended to configure SRAM_X2 to secure-privilege level. • INSYNC bit and input value updated in Table 523 “SCT configuration register (CONFIG, offset = 0x000)”. • In Figure 53 “SCT0 input multiplexing” of Chapter 18 “LPC55S6x/LPC55S2x/LPC552x Input Multiplexing (INPUTMUX)” updated SCTASYNCCLK PLL to read CGU SCTASYNCCLK. • FRG maximum allowed output frequency updated in Section 4.5.49 “Fractional rate divider for each Flexcomm Interface frequency”. • Added Section 9.3.4 “runBootloader API” in Chapter 9 “LPC55S6x/LPC55S2x/LPC552x Flash API”.
UM11126 v. 2.3	19 May 2021	Added DICE information in Table 38 “Register overview: SYSCON (base address = 0x50000000)”, Section 4.5.74 “DICE register”, and Section 48.9 “DICE”. Made RNG updates to Chapter 48 “LPC55S6x/LPC55S2x/LPC552x Security features”.
UM11126 v. 2.2	20 April 2021	Added footnotes to the end of Table 888 “Register overview: AHB_Secure_CTRL (base address = 0x500AC000)”, stating: • Unlike other register tables, the base address noted for this function is the Secure address. This is because these registers are configured by Secure code and Non-secure accesses are denied. • For all reserved registers and reserved bits within address range 0x500AC0F0 to 0x500AC174, value must be set to 1. See the SDK software platform for example code.
UM11126 v. 2.1	10 July 2020	Includes various updates based on Documentation Issues including Section 39.6.12 “Trigger control registers” update of FIFO_SEL_A/B and Section 4.5.61 “FMC configuration register” update of Flash memory access time (FLASHTIM) values.
UM11126 v. 2.0	1 May 2020	Includes various updates.
UM11126 v. 1.9	12 November 2019	Includes IAP APIs as attachment and other minor revisions and updates.
UM11126 v. 1.8	1 October 2019	Updates Section 39.7.6 “Temperature sensor” with new details. Includes a new section for power APIs Section 14.4.1 “POWER_SetVoltageForFreq (unit32_t system_freq_hz)”. Includes new content Section 2.1.8 “RAM configuration”, a new description for DC-DC conversions in Section 14.4 “Power related API descriptions”, and miscellaneous other modifications and additions.
UM11126 v. 1.7	30 August 2019	Include improvements for the Chapter 51 “Debug Subsystem” and other improvements and updates.
UM11126 v. 1.6	22 July 2019	Minor updates. Added device revision 1B and LPC55S2x/LPC552x derivatives.

Revision history *…continued*

Document ID	Release Date	Description
UM11126 v. 1.5	17 July 2019	Updated device revision information, added description for TXIGNORE bit in the FIFO write data register, updated SPI block diagram to include general controls.
UM11126 v. 1.4	15 June 2019	Includes additional updates. Added support for 150 MHz CPU frequency.
UM11126 v. 1.3	8 May 2019	Includes additional updates.
UM11126 v. 1.2	3 May 2019	Incorporates miscellaneous updates and improvements.
UM11126 v. 1.1	22 March 2019	LPC55S6x/LPC55S2x/LPC552x User Manual • Updated Ordering options table Table 2 “Ordering options”. • Made many updates to the Syscon chapter Chapter 4 “LPC55S6x/LPC55S2x/LPC552x SYSCON”.
UM11126 v.1.0	24 February 2019	Initial release. LPC55S6x/LPC55S2x/LPC552x User Manual.

1.1 Introduction

The LPC55S6x/LPC55S2x/LPC552x is an Arm Cortex®-M33 based micro-controller for embedded applications. These devices include up to 320 kB of on-chip SRAM, up to 640 kB on-chip flash, high-speed and full-speed USB host and device interface with crystal-less operation for full-speed, one SD/MMC/SDIO interface, five general-purpose timers, one SCTimer/PWM, one RTC/alarm timer, one 24-bit Multi-Rate Timer (MRT), a Windowed Watchdog Timer (WWDT), one high speed SPI (50 MHz), eight flexible serial communication peripherals (each of which can be a USART, SPI, I^2C, or I^2S interface), one 16-bit 1.0 Msamples/sec ADC capable of simultaneous conversions, temperature sensor.

The Arm Cortex M33 provides a security foundation, offering isolation to protect valuable IP and data with TrustZone® technology. It simplifies the design and software development of digital signal control systems with the integrated digital signal processing (DSP) instructions. To support security requirements, the LPC55S6x/ LPC55S2x also offers support for HASH, AES, RSA, UUID, DICE, dynamic encrypt and decrypt, debug authentication, and TBSA compliance.

1.2 Features

The following features are provided by the LPC55S6x/LPC55S2x/LPC552x family of parts. Depending on the processor, some features described here may or may not be available as noted. See Section 1.9.1 “Ordering options” for complete details.

- ARM Cortex-M33 core (CPU0, r0p3):
  - Running at a frequency of up to 150 MHz (device revision 1B only).
  - TrustZone, Floating Point Unit (FPU) and Memory Protection Unit (MPU).
  - ARM Cortex M33 built-in Nested Vectored Interrupt Controller (NVIC).
  - Integrated digital signal processing (DSP) instructions.
  - Non-maskable Interrupt (NMI) input with a selection of sources.
  - Serial Wire Debug with eight breakpoints and four watch points. Includes Serial Wire Output for enhanced debug capabilities.
  - System tick timer.
- ARM Cortex-M33 coprocessor (CPU1, r0p3):
  - Running at a frequency of up to 150 MHz (device revision 1B only).
  - The configuration of this instance does not include MPU, FPU, DSP, ETM, and Trustzone.
  - System tick timer.
- CASPER (not available on LPC552x) crypto co-processor that enables hardware acceleration for various functions required for certain asymmetric cryptographic algorithms, such as, Elliptic Curve Cryptography (ECC).

- PowerQuad hardware accelerator for (fixed and floating point unit) CMSIS DSP functions with support of software API faster execution of ARM CMSIS instruction set.
- On-chip memory:
  - Up to 640 kB on-chip flash program memory with flash accelerator and 512 byte page erase and write.
  - Up to 320 kB total SRAM consisting of 32 kB SRAM on Code Bus, 272 kB SRAM on System Bus (272 kB is contiguous), and additional 16 kB USB SRAM on System Bus which can be used by the USB interface or for general purpose use.
- PRINCE (not available on LPC552x) module for real-time encryption of data being written to on-chip flash and decryption of encrypted flash data during read to allow asset protection, such as securing application code, and enabling secure flash update.
- On-chip ROM bootloader supports:
  - Booting of images from on-chip flash.
  - Supports CRC32 image integrity checking.
  - Supports flash programming through In System Programming (ISP) commands over following interfaces: USB0/1 interfaces using HID class device, UART interface (Flexcomm 0) with auto baud, SPI slave interfaces (Flexcomm 3) using mode 3 (CPOL = 1 and CPHA = 1), and I2C slave interface (Flexcomm 1)
  - ROM API functions: Flash programming API, Power control API, and Secure firmware update API using NXP Secure Boot file format, version 2.0 (SB2 files).
  - Supports booting of images from PRINCE encrypted flash regions.
  - Support NXP Debug Authentication Protocol version 1.0 (RSA-2048) and 1.1 (RSA-4096).
  - Supports setting a sealed part to Fault Analysis mode through Debug authentication.
- Secure Boot support:
  - Uses RSASSA-PKCS1-v1_5 signature of SHA256 digest as cryptographic signature verification.
  - Supports RSA-2048 bit public keys (2048 bit modulus, 32-bit exponent).
  - Supports RSA-4096 bit public keys (4096 bit modulus, 32-bit exponent).
  - Uses x509 certificate format to validate image public keys.
  - Supports up to four revocable Root of Trust (RoT) or Certificate Authority keys, Root of Trust (RoT) establishment by storing the SHA-256 hash digest of the hashes of four RoT public keys in protected flash region (PFR).
  - Supports anti-rollback feature using image key revocation and supports up to 16 Image key certificates revocations using Serial Number field in x509 certificate.
- Serial interfaces:
  - Flexcomm Interface contains up to nine serial peripherals (Flexcomm Interface 0-7 and Flexcomm Interface 8). Each Flexcomm Interface (except Flexcomm 8, which is dedicated for high-speed SPI) can be selected by software to be a USART, SPI, I^2C, and I^2S interface. Each Flexcomm Interface includes a FIFO that supports

USART, SPI, and I^2S. A variety of clocking options are available to each Flexcomm Interface, including a shared fractional baud-rate generator, and time-out feature.Flexcomm interfaces 0 to 7 each provide one channel pair of I^2S.

- I^2C-bus interfaces support Fast-mode and Fast-mode Plus with data rates of up to 1Mbit/s and with multiple address recognition and monitor mode. Two sets of true I^2C pads also support high-speed mode (3.4 Mbit/s) as a slave.
- USB 2.0 full speed host/device controller with on-chip PHY and dedicated DMA controller supporting crystal-less operation in device mode using software library example in technical note TN00063.
- USB 2.0 high-speed host/device controller with on-chip high-speed PHY.

- Digital peripherals:
  - DMA0 controller with 23 channels and up to 22 programmable triggers, able to access all memories and DMA-capable peripherals.
  - DMA1 controller with 10 channels and up to 15 programmable triggers, able to access all memories and DMA-capable peripherals.
  - Secured digital input/output (SD/MMC and SDIO) card interface with DMA support. SDIO with support for up to two cards. Supported card types are MMC, SDIO, and CE-ATA. Supports SD2.0, and SDR25 (52MHz).
  - CRC engine block can calculate a CRC on supplied data using one of three standard polynomials with DMA support.
  - Up to 64 General-Purpose Input/Output (GPIO) pins.
  - GPIO registers are located on the AHB for fast access. The DMA supports GPIO ports.
  - Up to eight GPIOs can be selected as Pin Interrupts (PINT), triggered by rising, falling or both input edges.
  - Two GPIO grouped interrupts (GINT) enable an interrupt based on a logical (AND/OR) combination of input states.
  - I/O pin configuration with support for up to 16 function options.
  - Programmable Logic Unit (PLU) to create small combinatorial and/or sequential logic networks including state machines.

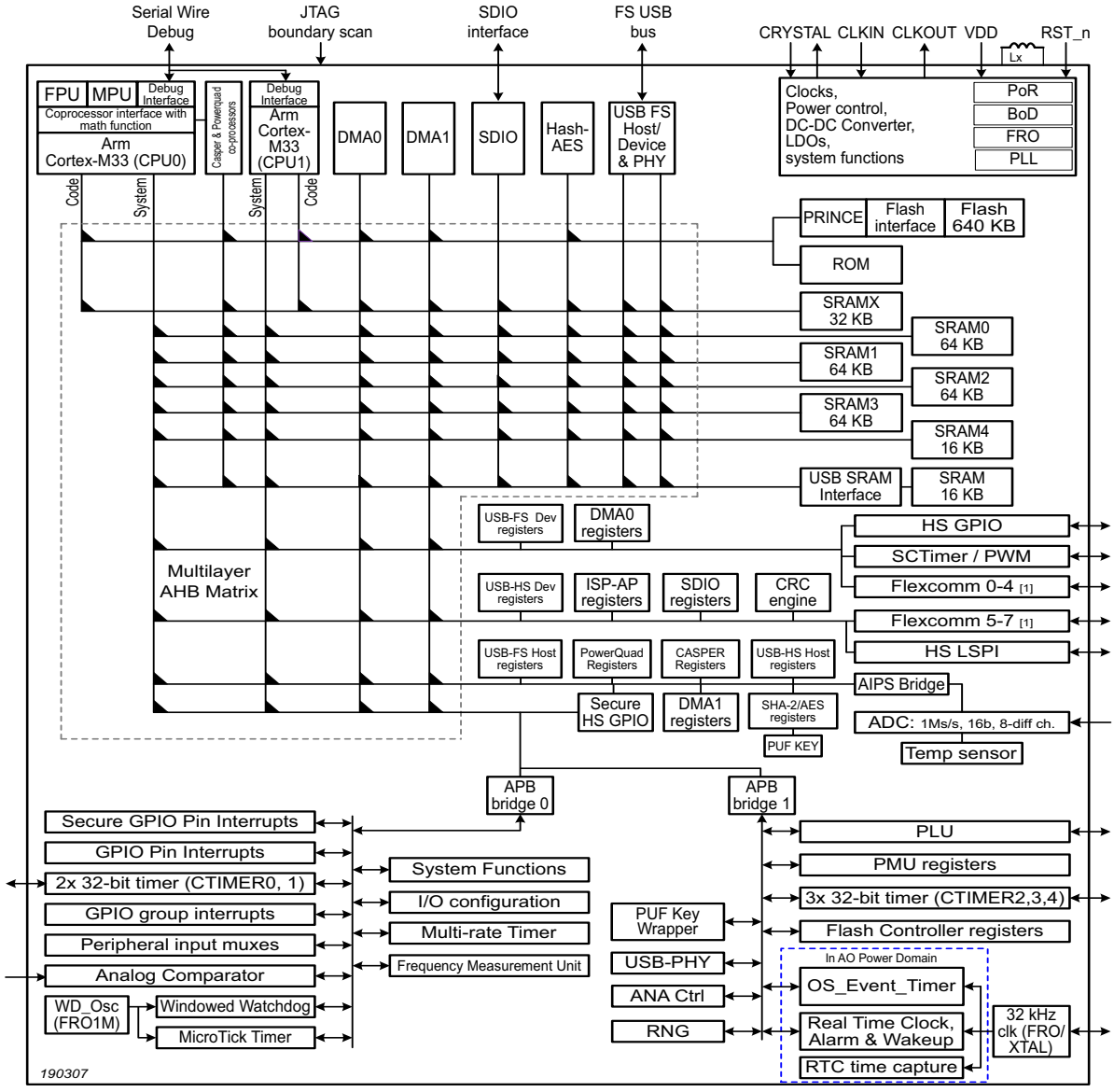
- Security features:
  - ARM TrustZone enabled (not available on LPC552x).
  - AES-256 encryption/decryption engine with keys fed directly from PUF or a software supplied key (not available on LPC552x).
  - Secure Hash Algorithm (SHA2) module supports secure boot with dedicated DMA controller (not available on LPC552x).
  - Physical Unclonable Function (PUF) using dedicated SRAM for silicon fingerprint. PUF can generate, store, and reconstruct key sizes from 64-bits to 4096-bits. Includes hardware for key extraction (not available on LPC552x).
  - True Random Number Generator (TRNG).
  - 128-bit unique device serial number for identification (UUID).
  - Secure GPIO.

- Timers:
  - Five 32-bit standard general purpose asynchronous timers/counters, which support up to four capture inputs and four compare outputs, PWM mode, and external count input. Specific timer events can be selected to generate DMA requests.
  - One SCTimer/PWM with eight input and ten output functions (including 16 capture and match registers). Inputs and outputs can be routed to or from external pins and internally to or from selected peripherals. Internally, the SCTimer/PWM supports 16 captures/matches, 16 events, and 32 states.
  - 32-bit Real-time Clock (RTC) with 1s resolution running in the always-on power domain. Another timer in the RTC can be used for wake-up from all low power modes including deep-power down, with 1ms resolution. The RTC is clocked by the 32 kHz FRO or 32.768 kHz external crystal.
  - Multiple-channel multi-rate 24-bit timer (MRT) for repetitive interrupt generation at up to four programmable, fixed rates.
  - Windowed Watchdog Timer (WWDT) with FRO 1 MHz as clock source.
  - The Micro-Tick Timer running from the watchdog oscillator can be used to wake-up the device from sleep and deep-sleep modes. Includes 4 capture registers with pin inputs.
  - 42-bit free running OS Timer as continuous time-base for the system, available in any reduced power modes. It runs on 32kHz clock source, allowing a count period of more than four years.
- Analog peripherals
  - 16-bit ADC with five differential channel pair (or 10 single-ended channels), and with multiple internal and external trigger inputs and sample rates of up to 1.0 MSamples/sec. The ADC supports simultaneous conversions on two ADC input channels belonging to a differential pair.
  - Integrated temperature sensor connected to the ADC.
  - Comparator with five input pins and external or internal reference voltage.
- Clock generation
  - Internal Free Running Oscillator (FRO). This oscillator provides a selectable 96 MHz output, and a 12 MHz output (divided down from the selected higher frequency) that can be used as a system clock. The FRO is trimmed to +/- 2% accuracy over the entire voltage and temperature range.
  - 32 kHz FRO. The FRO is trimmed to +/- 2% accuracy over the entire voltage and temperature range.
  - Internal low power oscillator (FRO 1 MHz) trimmed to +/- 15% accuracy over the entire voltage and temperature range.
  - Crystal oscillator with an operating frequency of 16 MHz to 32 MHz. Option for external clock input (bypass mode) for clock frequencies of up to 25 MHz.
  - Crystal oscillator with 32.768 kHz operating frequency. Option for external clock input (bypass mode) for clock frequencies of up to 100 kHz.

  - PLL0 and PLL1 allows CPU operation up to the maximum CPU rate without the need for a high-frequency external clock. PLL0 and PLL1 can run from the internal FRO 12 MHz output, the external oscillator, internal FRO 1 MHz output, or the 32.768 kHz RTC oscillator.
  - Clock output function with divider to monitor internal clocks.
  - Frequency measurement unit for measuring the frequency of any on-chip or off-chip clock signal.
  - Each crystal oscillator has one embedded capacitor bank, where each can be used as an integrated load capacitor for the crystal oscillators. Using APIs, the capacitor banks on each crystal pin can tune the frequency for crystals with a Capacitive Load (CL), thus conserving board space and reducing costs.

- Power-saving modes and wake-up:
  - Integrated PMU (Power Management Unit) to minimize power consumption.
  - Reduced power modes: Sleep, deep-sleep with RAM retention, power-down with RAM retention and CPU0 retention, and deep power-down with RAM retention.
  - Configurable wake-up options from peripherals interrupts.
  - The Micro-Tick Timer running from the watchdog oscillator, and the Real-Time Clock (RTC) running from the 32.768 kHz clock, can be used to wake-up the device from sleep and deep-sleep modes.
  - Power-On Reset (POR).
  - Brown-Out Detectors (BOD) for VBAT_DCDC for forced reset or interrupt.
- Operating from internal DC-DC converter.
- Single power supply 1.8 V to 3.6 V.
- JTAG boundary scan supported.
- Operating temperature range −40 °C to +105 °C.
- Available in HLQFP100, HTQFP64, VFBGA98, and VFBGA59 packages.

1.3 Block diagram



Notes:

[1] : Each FlexComm includes USART, SPI, I2C and I2S functions. Flexcomms 0 to 7 each provide 1 channel-pair of I2S function.

Fig 1. LPC55S6x Block diagram

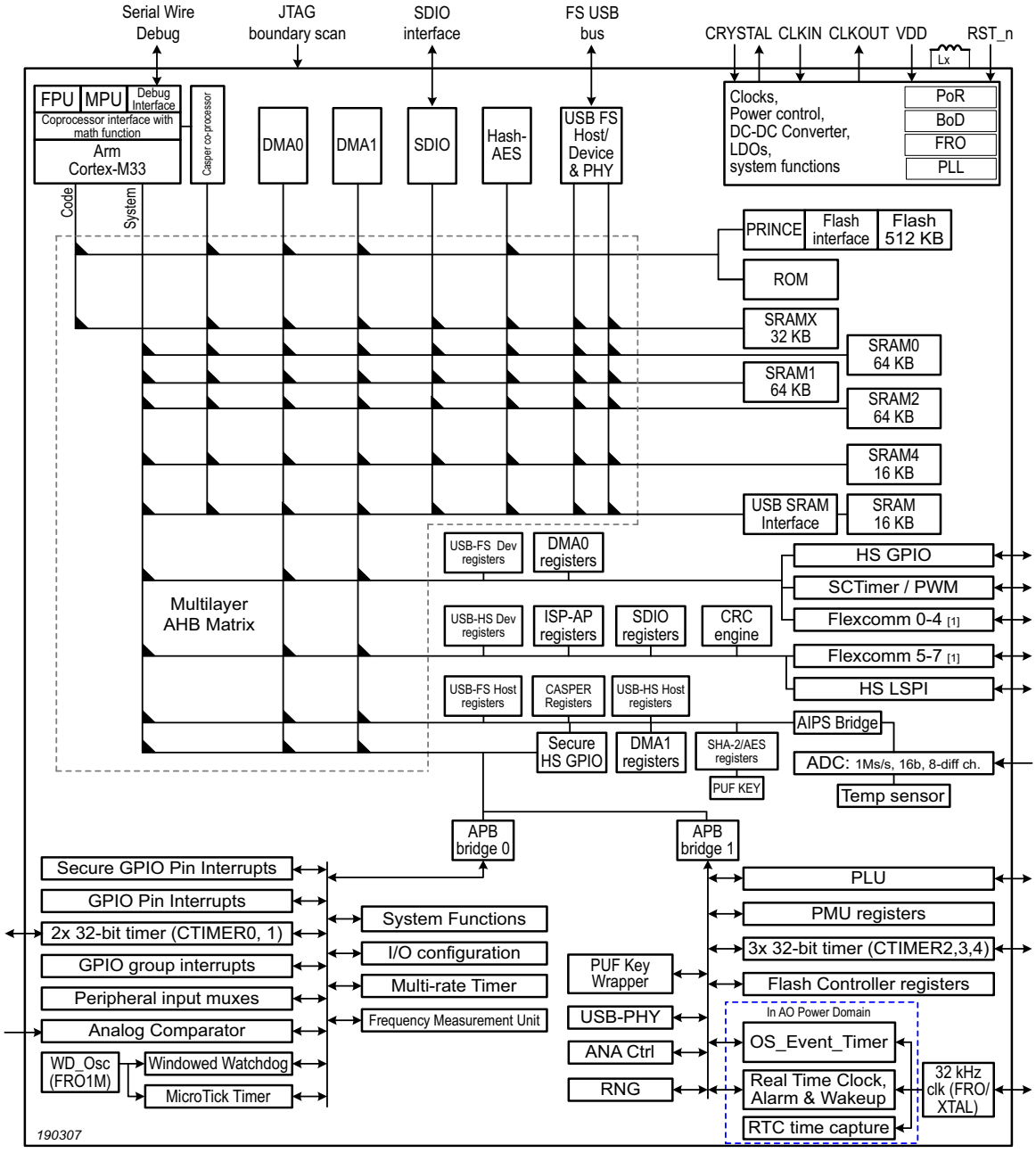


Fig 2. LPC55S2x Block Diagram

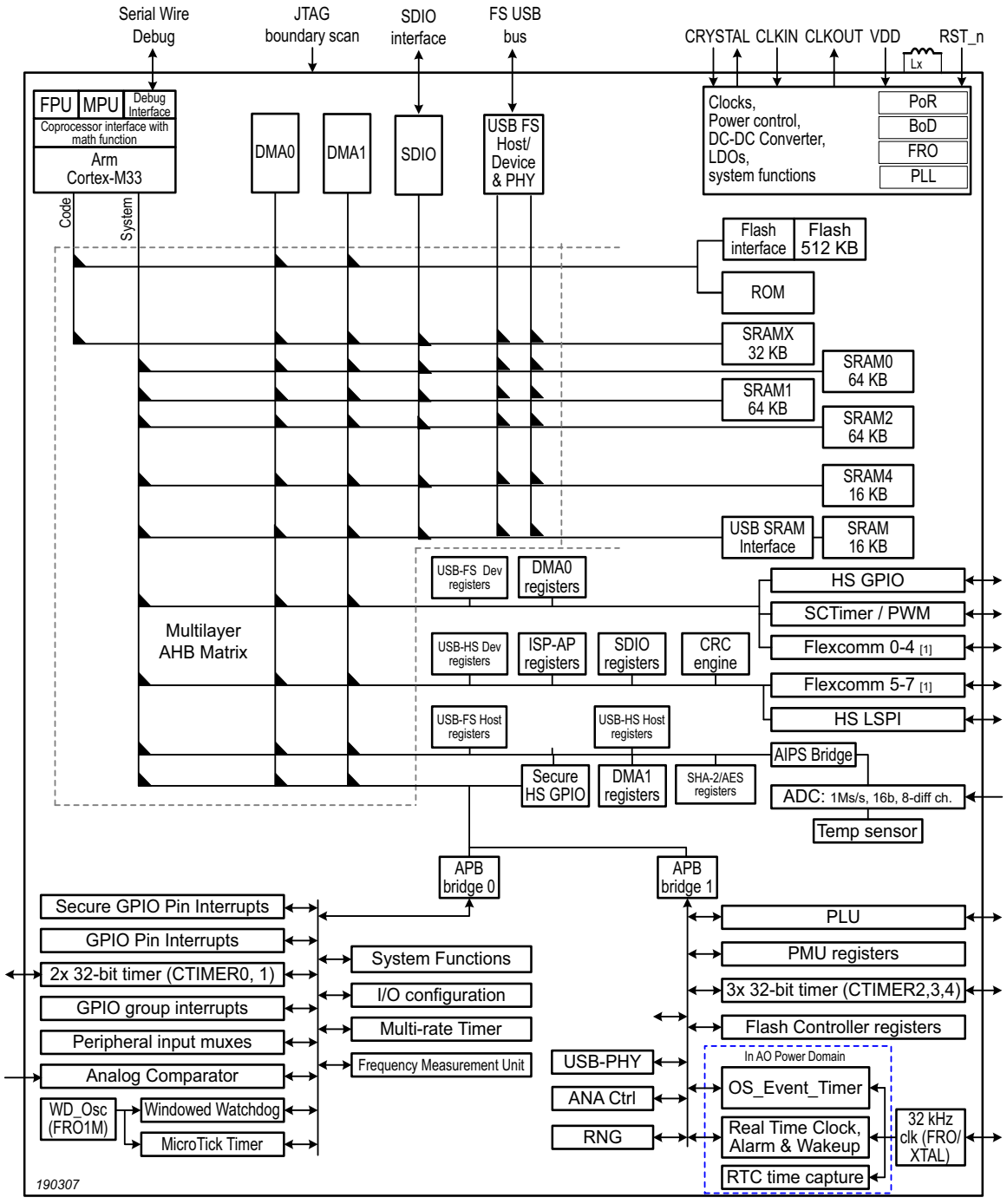


Fig 3. LPC552x Block diagram

1.4 Architectural overview

The Arm Cortex M33 includes two AHB-Lite buses, one system bus and one code and bus. The Code AHB (C-AHB) interface is used for any instruction fetch and data access to the Code region of the ARMv8-M memory map ([0x00000000 - 0x1FFFFFFF]). The System AHB (S-AHB) interface is used for instruction fetch and data access to all other regions of the ARMv8-M memory map ([0x20000000 - 0xFFFFFFFF]).

The LPC55S6x/LPC55S2x/LPC552x uses a multi-layer AHB matrix to connect the Arm Cortex M33 buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing peripherals that are on different slave ports of the matrix to be accessed simultaneously by different bus masters.

1.5 Arm Cortex-M33 TrustZone

The Arm Cortex-M33 is a general purpose, 32-bit microprocessor, which offers high performance and very low power consumption. The Arm Cortex M33 offers many new features, including a Thumb-2 instruction set, low interrupt latency, hardware multiply and divide, interruptable/continuable multiple load and store instructions, automatic state save and restore for interrupts, tightly integrated interrupt controller with wake-up interrupt controller, and multiple core buses capable of simultaneous accesses.

A 3-stage pipeline is employed so that all parts of the processing and memory systems can operate continuously. Typically, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory.

The Arm Cortex-M33 provides a security foundation, offering isolation to protect valuable IP and data with TrustZone technology. It simplifies the design and software development of digital signal control systems with the integrated digital signal processing (DSP) instructions.

1.6 Arm Cortex-M33 integrated Floating Point Unit (FPU)

The FPU fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions.
The FPU provides floating-point computation functionality that is compliant with the ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic, referred to as the IEEE 754 standard.

1.7 Arm Cortex-M33 (CPU1)

The LPC55S6x/LPC55S2x/LPC552x device includes a second instance of Cortex M33. The configuration of this instance does not include MPU, FPU, DSP, ETM, Trustzone (SECEXT), Secure Attribution Unit (SAU) or co-processor interface. It supports the same debug levels and interrupt lines as the primary CPU.

1.8 On-chip Static RAM

The LPC55S6x/LPC55S2x/LPC552x support various configurations of SRAM with separate bus master access for higher throughput and individual power control for low-power operation. See the appropriate block diagram for more details.

1.9 Ordering information

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
LPC55S66JBD100	HLQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 0.5mm pitch	SOT1570-3
LPC55S69JBD100	HLQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 0.5mm pitch	SOT1570-3
LPC55S66JEV98	VFBGA98	thin fine-pitch ball grid array package; 98 balls; body 7‘ 7‘ 0.5 mm	SOT1982-1
LPC55S69JEV98	VFBGA98	thin fine-pitch ball grid array package; 98 balls; body 7‘ 7‘ 0.5 mm	SOT1982-1
LPC55S66JBD64	HTQFP64	thin fine-pitch ball grid array package; 64 leads; body 10 × 10 × 0.5mm pitch	SOT855-5
LPC55S69JBD64	HTQFP64	thin fine-pitch ball grid array package; 64 leads; body 10 × 10 × 0.5mm pitch	SOT855-5
LPC55S28JBD100	HLQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 0.5mm pitch	SOT1570-3
LPC55S26JBD100	HLQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 0.5mm pitch	SOT1570-3
LPC55S28JEV98	VFBGA98	thin fine-pitch ball grid array package; 98 balls; body 7‘ 7‘ 0.5 mm	SOT1982-1
LPC55S26JEV98	VFBGA98	thin fine-pitch ball grid array package; 98 balls; body 7‘ 7‘ 0.5 mm	SOT1982-1
LPC55S28JBD64	HTQFP64	thin fine-pitch ball grid array package; 64 leads; body 10 × 10 × 0.5mm pitch	SOT855-5
LPC55S26JBD64	HTQFP64	thin fine-pitch ball grid array package; 64 leads; body 10 × 10 × 0.5mm pitch	SOT855-5
LPC5528JBD100	HLQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 0.5mm pitch	SOT1570-3
LPC5526JBD100	HLQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 0.5mm pitch	SOT1570-3
LPC5528JEV98	VFBGA98	thin fine-pitch ball grid array package; 98 balls; body 7‘ 7‘ 0.5 mm	SOT1982-1
LPC5526JEV98	VFBGA98	thin fine-pitch ball grid array package; 98 balls; body 7‘ 7‘ 0.5 mm	SOT1982-1
LPC5528JBD64	HTQFP64	thin fine-pitch ball grid array package; 64 leads; body 10 × 10 × 0.5mm pitch	SOT855-5
LPC5526JBD64	HTQFP64	thin fine-pitch ball grid array package; 64 leads; body 10 × 10 × 0.5mm pitch	SOT855-5
LPC55S28JEV59	VFBGA59	Thin fine-pitch ball grid array package; 59 balls; body 4 × 4 × 0.4 mm pitch	SOT2162-1
LPC5528JEV59	VFBGA59	Thin fine-pitch ball grid array package; 59 balls; body 4 × 4 × 0.4 mm pitch	SOT2162-1
LPC55S69JEV59	VFBGA59	Thin fine-pitch ball grid array package; 59 balls; body 4 × 4 × 0.4 mm pitch	SOT2162-1
LPC55S66JEV59	VFBGA59	Thin fine-pitch ball grid array package; 59 balls; body 4 × 4 × 0.4 mm pitch	SOT2162-1

1.9.1 Ordering options

Table 2. Ordering options

Type number	Max CPU Frequency (MHz)	Primary core (CPU0)	Secondary core (CPU1)	Power Quad	CASPER	Flash/KB	Total SRAM/KB	Secure boot	TrustZone	PRINCE	PUF Controller	HASH-AES	SDIO	USB	GPIO
LPC55S66JBD100	150 [1]	yes	yes	yes	yes	256	144	yes	yes	yes	yes	yes	yes	FS + HS	64
LPC55S69JBD100	150 [1]	yes	yes	yes	yes	640	320	yes	yes	yes	yes	yes	yes	FS + HS	64
LPC55S66JEV98	150 [1]	yes	yes	yes	yes	256	144	yes	yes	yes	yes	yes	yes	FS + HS	64
LPC55S69JEV98	150 [1]	yes	yes	yes	yes	640	320	yes	yes	yes	yes	yes	yes	FS + HS	64
LPC55S66JBD64	150 [1]	yes	yes	yes	yes	256	144	yes	yes	yes	yes	yes	yes	FS + HS	36
LPC55S69JBD64	150 [1]	yes	yes	yes	yes	640	320	yes	yes	yes	yes	yes	yes	FS + HS	36
LPC55S28JBD100	150	yes	-	-	yes	512	256	yes	-	yes	yes	yes	yes	FS + HS	64
LPC55S26JBD100	150	yes	-	-	yes	256	144	yes	-	yes	yes	yes	yes	FS + HS	64
LPC55S28JEV98	150	yes	-	-	yes	512	256	yes	-	yes	yes	yes	yes	FS + HS	64
LPC55S26JEV98	150	yes	-	-	yes	256	144	yes	-	yes	yes	yes	yes	FS + HS	64
LPC55S28JBD64	150	yes	-	-	yes	512	256	yes	-	yes	yes	yes	yes	FS + HS	36
LPC55S26JBD64	150	yes	-	-	yes	256	144	yes	-	yes	yes	yes	yes	FS + HS	36
LPC5528JBD100	150	yes	-	-	-	512	256	-	-	-	-	-	yes	FS + HS	64
LPC5526JBD100	150	yes	-	-	-	256	144	-	-	-	-	-	yes	FS + HS	64
LPC5528JEV98	150	yes	-	-	-	512	256	-	-	-	-	-	yes	FS + HS	64
LPC5526JEV98	150	yes	-	-	-	256	144	-	-	-	-	-	yes	FS + HS	64
LPC5528JBD64	150	yes	-	-	-	512	256	-	-	-	-	-	yes	FS + HS	36
LPC5526JBD64	150	yes	-	-	-	256	144	-	-	-	-	-	yes	FS + HS	36
LPC55S28JEV59	150	yes	-	-	yes	512	256	yes	-	yes	yes	yes	yes	HS	37
LPC5528JEV59	150	yes	-	-	-	512	256	-	-	-	-	-	yes	HS	37
LPC55S69JEV59	150 [1]	yes	yes	yes	yes	640	320	yes	yes	yes	yes	yes	yes	HS	37
LPC55S66JEV59	150 [1]	yes	yes	yes	yes	256	144	yes	yes	yes	yes	yes	yes	HS	37

[1] Device revision 1B operates at a maximum CPU frequency of up to 150 MHz. Device revision 0A operates at a maximum CPU frequency of up to 100 MHz.

2.1 General description

2.1.1 AHB multilayer matrix

The LPC55S6x/LPC55S2x/LPC552x uses a multi-layer AHB matrix to connect the CPU buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing peripherals that are on different slave ports of the matrix to be accessed simultaneously by different bus masters. The device block diagram in Figure 1 shows details of the available matrix connections.

2.1.2 Memory Protection Unit (MPU)

CPU0 has a memory protection unit (MPU) that provides fine grain memory control, enabling applications to implement security privilege levels, separating code, data and stack on a task-by-task basis. Such requirements are critical in many embedded applications.

The MPU register interface is located on the CPU private peripheral bus.

2.1.3 TrustZone and system mapping on this device

The implementation of ARM TrustZone for CPU0 involves using address bit 28 to divide the address space into potential secure and non-secure regions. Address bit 28 is not decoded in memory access hardware, so each physical location appears in two places on whatever bus they are located on. Other hardware determines which kinds of accesses (including non-secure callable) are actually allowed for any particular address.

Table 3 shows the overall mapping of the code and data buses for secure and non-secure accesses to various device resources.

Remark: Address regions considered secure by TrustZone may also be accessible to CPU1 if it is assigned as a secure master and marked as secure by checker hardware.

Remark: In the peripheral description chapters of this manual, only the native (non-secure) base address is noted, secure base addresses can be found in this chapter or created by setting bit 28 in the address as needed.

Table 3. TrustZone and system general mapping

Start address	End address	TrustZone, CPU0 only	CPU bus	CM-33 usage (both CPUs)
0x0000 0000	0x0FFF FFFF	Non-secure	Code	Flash memory, Boot ROM, SRAM X.
0x1000 0000	0x1FFF FFFF	Secure	Code	Same as above.
0x2000 0000	0x2FFF FFFF	Non-secure	Data	SRAM 0, SRAM 1, SRAM 2, SRAM 3, SRAM 4.
0x3000 0000	0x3FFF FFFF	Secure	Data	Same as above.
0x4000 0000	0x4FFF FFFF	Non-secure	Data	AHB and APB peripherals.
0x5000 0000	0x5FFF FFFF	Secure	Data	Same as above.

[1] The size shown for peripherals spaces indicates the space allocated in the memory map, not the actual space used by the peripheral or memory.

[2] Selected areas of secure regions may be marked as non-secure callable.

2.1.4 Links to specific memory map descriptions and tables:

- Section 2.1.5 “Memory map overview”
- Section 2.1.6 “APB peripherals”
- Section 2.1.7 “AHB peripherals”

2.1.5 Memory map overview

Table 4 gives a more detailed memory map as seen by the 2 Cortex-M33 (both CPU0 and CPU1). The purpose of the four address spaces for the shared RAMs is outlined at the beginning of this chapter. The details of which shared RAM regions are on which AHB matrix slave ports can be seen here.

Table 4. Memory map overview

AHB port	Non-secure start address	Non-secure end address	Secure start address	Secure end address	Function [1]
0	0x0000 0000	0x0009 FFFF	0x1000 0000	0x1009 FFFF	Flash memory, on CM33 code bus. The last 16 pages (10 KB) are reserved on the 640 KB flash devices resulting in 630 KB internal flash memory.
	0x0300 0000	0x0301 FFFF	0x1300 0000	0x1301 FFFF	Boot ROM, on CM33 code bus.
1	0x0400 0000	0x0400 7FFF	0x1400 0000	0x1400 7FFF	SRAM X on CM33 code bus, 32 KB. SRAMX_0 (0x1400 0000 to 0x1400 0FFF) and SRAMX_1 (0x1400 4000 to 0x1400 4FFF) are used for Casper (total 8 KB). If CPU retention used in power-down mode, SRAMX_2 (0x1400 6000 to 0x1400 65FF) is used (total 1.5 KB) by default in power API and this is user configurable within SRAMX_2 and SRAMX_3.
2	0x2000 0000	0x2000 FFFF	0x3000 0000	0x3000 FFFF	SRAM 0 on CM33 data bus, 64 KB.
3	0x2001 0000	0x2001 FFFF	0x3001 0000	0x3001 FFFF	SRAM 1 on CM33 data bus, 64 KB.
4	0x2002 0000	0x2002 FFFF	0x3002 0000	0x3002 FFFF	SRAM 2 on CM33 data bus, 64 KB.
5	0x2003 0000	0x2003 FFFF	0x3003 0000	0x3003 FFFF	SRAM 3 on CM33 data bus, 64 KB.
6	0x2004 0000	0x2004 3FFF	0x3004 0000	0x3004 3FFF	SRAM 4 on CM33 data bus, 16 KB. Entire SRAM 4 is used by PowerQuad when PowerQuad is enabled.
7	0x4000 0000	0x4001 FFFF	0x5000 0000	0x5001 FFFF	AHB to APB bridge 0. See Section 2.1.6.
	0x4002 0000	0x4003 FFFF	0x5002 0000	0x5003 FFFF	AHB to APB bridge 1. See Section 2.1.6.
8	0x4008 0000	0x4008 FFFF	0x5008 0000	0x5008 FFFF	AHB peripherals. See Section 2.1.7.
9	0x4009 0000	0x4009 FFFF	0x5009 0000	0x5009 FFFF	AHB peripherals. See Section 2.1.7.
10	0x400A 0000	0x400A FFFF	0x500A 0000	0x500A FFFF	AHB peripherals. See Section 2.1.7.
11	0x4010 0000	0x4010 FFFF	0x5010 0000	0x5010 FFFF	AHB peripherals. See Section 2.1.7.

[1] Gaps between AHB matrix slave ports are not shown.

2.1.6 APB peripherals

Table 5 provides details of the addresses for APB peripherals. APB peripherals have both secure and non-secure access possibilities.

Table 5. APB peripherals memory map

APB bridge	Non-secure base address	Secure base address	Peripheral
0	0x4000 0000	0x5000 0000	Syscon.
	0x4000 1000	0x5000 1000	IOCON. Pin function selection and pin control setup.
	0x4000 2000	0x5000 2000	Group GPIO input interrupt 0 (GINT0).
	0x4000 3000	0x5000 3000	Group GPIO input interrupt 1 (GINT1).
	0x4000 4000	0x5000 4000	Pin interrupt and pattern match (PINT).
	0x4000 5000	0x5000 5000	Secure pin interrupt and pattern match.
	0x4000 6000	0x5000 6000	Input multiplexing 0 and frequency measure.
	0x4000 7000	0x5000 7000	Reserved.
	0x4000 8000	0x5000 8000	CTimer0 (standard counter/timer 0).
	0x4000 9000	0x5000 9000	CTimer1 (standard counter/timer 1).
	0x4000 C000	0x5000 C000	WWDT0 (windowed watchdog timer 0).
	0x4000 D000	0x5000 D000	MRT (Multi-Rate Timer).
	0x4000 E000	0x5000 E000	Utick (micro-tick timer).
	0x4001 3000	0x5001 3000	Analog controls.
	0x4001 5000	0x5001 5000	Reserved.
1	0x4002 3000	0x5002 3000	Sysctl (I^2S signal sharing).
	0x4002 8000	0x5002 8000	CTimer2 (standard counter/timer 2).
	0x4002 9000	0x5002 9000	CTimer3 (standard counter/timer 3).
	0x4002 A000	0x5002 A000	CTimer4 (standard counter/timer 4).
	0x4002 C000	0x5002 C000	RTC & Wake-up timer.
	0x4002 D000	0x5002 D000	OS_Event Timer.
	0x4003 4000	0x5003 4000	Flash controller.
	0x4003 5000	0x5003 5000	PRINCE dynamic encrypt/decrypt
	0x4003 8000	0x5003 8000	USB HS Phy.
	0x4003 A000	0x5003 A000	True Random Number Generator.
	0x4003 B000	0x5003 B000	PUF (Physical Unclonable Function).
	0x4003 D000	0x5003 D000	PLU (Programmable Logic Unit).

2.1.7 AHB peripherals

Table 6 provides details of the addresses for AHB peripherals. AHB peripherals have both secure and non-secure access possibilities.

Table 6. AHB peripheral memory map

AHB port	Non-secure base address	Secure base address	Peripheral
8	0x4008 2000	0x5008 2000	DMA0 registers.
	0x4008 4000	0x5008 4000	FS USB device registers.
	0x4008 5000	0x5008 5000	SCTimer/PWM.
	0x4008 6000	0x5008 6000	Flexcomm Interface 0.
	0x4008 7000	0x5008 7000	Flexcomm Interface 1.
	0x4008 8000	0x5008 8000	Flexcomm Interface 2.
	0x4008 9000	0x5008 9000	Flexcomm Interface 3.
	0x4008 A000	0x5008 A000	Flexcomm Interface 4.
	0x4008 B000	0x5008 B000	Inter-CPU Mailbox.
	0x4008 C000	0x5008 C000	High-Speed GPIO.
9	0x4009 4000	0x5009 4000	HS USB device registers.
	0x4009 5000	0x5009 5000	CRC Engine.
	0x4009 6000	0x5009 6000	Flexcomm Interface 5.
	0x4009 7000	0x5009 7000	Flexcomm Interface 6.
	0x4009 8000	0x5009 8000	Flexcomm Interface 7.
	0x4009 B000	0x5009 B000	SDIO registers.
	0x4009 C000	0x5009 C000	Debug Mailbox (DM-AP).
	0x4009 F000	0x5009 F000	High Speed SPI.
10	0x400A 0000	0x500A 0000	ADC0.
	0x400A 2000	0x500A 2000	FS USB Host registers.
	0x400A 3000	0x500A 3000	HS USB Host registers.
	0x400A 4000	0x500A 4000	Hash-AES registers.
	0x400A 5000	0x500A 5000	Casper
	0x400A 6000	0x500A 6000	PowerQuad
	0x400A 7000	0x500A 7000	DMA1 registers.
	0x400A 8000	0x500A 8000	Secure HS GPIO.
	0x400A C000	0x500A C000	Security control registers.
11	0x4010 0000	0x5010 0000	USB SRAM.

2.1.8 RAM configuration

Table 7 describes the RAM configuration for the LPC55S6x/LPC55S2x/LPC552x.

Table 7. RAM Configuration

RAM Total	RAM-X (KB)	RAM0 (KB)	RAM1 (KB)	RAM2 (KB)	RAM3 (KB)	RAM4 (KB)	USB-RAM (KB)
320 KB devices	32	64	64	64	64	16	16
256 KB devices	32	64	64	64	-	16	16
144 KB devices	32	64	32	-	-	-	16

3.1 How to read this chapter

Available interrupt sources may vary with specific LPC55xx device types.

3.2 Features

- Nested Vectored Interrupt Controller (NVIC) is an integral part of each CPU.
- Tightly coupled interrupt controller provides low interrupt latency.
- Controls system exceptions and peripheral interrupts.
- The NVIC of the Cortex-M33 supports:
  - 64 vectored interrupt slots.
  - Eight programmable interrupt priority levels with hardware priority level masking.
  - Vector table offset register VTOR.
  - Software interrupt generation.
  - Support for NMI from any interrupt, see Section 21.4.3.

3.3 General description

The tight coupling to the NVIC to the CPU allows for low interrupt latency and efficient processing of late arriving interrupts.

3.3.1 Interrupt sources

Table 8 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. The interrupt number does not imply any interrupt priority when interrupts are not given the same priority. In the case, where two interrupts for example are given the same priority, then the interrupt number below is relevant.

See Ref. 1 “Cortex-M33 DGUG” for detailed descriptions of the NVIC and the NVIC registers.

Table 8. Connection of interrupt sources to the NVIC

Interrupt	Name	Interrupt description	Flags
0	WDT BOD FLASH	Windowed watchdog timer, Brown Out Detect, Flash controller.	WARNINT - watchdog warning interrupt BODINTVAL - BOD interrupt level.
1	SDMA0	SDMA0 controller.	Interrupt A and interrupt B, error interrupt.
2	GPIO_GLOBALINT0	GPIO group 0.	Enabled pin interrupts.
3	GPIO_GLOBALINT1	GPIO group 1.	Enabled pin interrupts.

Table 8. Connection of interrupt sources to the NVIC *…continued*

Interrupt	Name	Interrupt description	Flags
4	GPIO_INT0_IRQ0	Pin interrupt 0 or pattern match engine slice 0.	PSTAT - pin interrupt status.
5	GPIO_INT0_IRQ1	Pin interrupt 1 or pattern match engine slice 1.	PSTAT - pin interrupt status.
6	GPIO_INT0_IRQ2	Pin interrupt 2 or pattern match engine slice 2.	PSTAT - pin interrupt status.
7	GPIO_INT0_IRQ3	Pin interrupt 3 or pattern match engine slice 3.	PSTAT - pin interrupt status.
8	UTICK	Micro-tick timer.	INTR.
9	MRT	Multi-rate timer.	Global MRT interrupts: GFLAG0, 1, 2, 3.
10	CTIMER0	Standard counter/timer CTIMER0.	Match and capture interrupts.
11	CTIMER1	Standard counter/timer CTIMER1.	Match and capture interrupts.
12	SCT	SCTimer.	EVFLAG SCT event.
13	CTIMER3	Standard counter/timer CTIMER3.	Match and capture interrupts.
14	Flexcomm Interface 0	Flexcomm Interface 0 (USART, SPI, I^2C, I^2S).	See enable read and set register of this module
15	Flexcomm Interface 1	Flexcomm Interface 1 (USART, SPI, I^2C, I^2S).	Same as Flexcomm0.
16	Flexcomm Interface 2	Flexcomm Interface 2 (USART, SPI, I^2C, I^2S)	Same as Flexcomm0
17	Flexcomm Interface 3	Flexcomm Interface 3 (USART, SPI, I^2C, I^2S)	Same as Flexcomm0.
18	Flexcomm Interface 4	Flexcomm Interface 4 (USART, SPI, I^2C, I^2S)	Same as Flexcomm0
19	Flexcomm Interface 5	Flexcomm Interface 5 (USART, SPI,I^2C, I^2S).	Same as Flexcomm0.
20	Flexcomm Interface 6	Flexcomm Interface 6 (USART, SPI, I^2C, I^2S).	Same as Flexcomm0.
21	Flexcomm Interface 7	Flexcomm Interface 7 (USART, SPI, I^2C, I^2S).	Same as Flexcomm0.
22	ADC	ADC0 completion.	See enable read and set register of this module.
23	Reserved	-	-
24	ACMP	Comparator Sub-system.	See enable read and set register of this module.
25	Reserved	-	-
26	Reserved	-	-
27	USB0_NEEDCLK	USB0 activity Interrupt.	See enable read and set register of this module.
28	USB0	USB0 host and device.	See enable read and set register of this module.
29	RTC	RTC alarm and wake-up interrupts.	See enable read and set register of this module.
30	Reserved	-	-
31	WAKEUP_IRQn or MAILBOX	Wakeup for low power mode (Power Down) use when wakeupio is enabled during power down. System IRQ for Mailbox	See Section 13.4.11 “Wake-up I/O cause register” in “LPC55S6x/LPC55S2x/LPC552x Power Management”.
32	GPIO_INT0_IRQ4	Pin interrupt 4 or pattern match engine slice 4 int	PSTAT - pin interrupt status.
33	GPIO_INT0_IRQ5	Pin interrupt 5 or pattern match engine slice 5 int	PSTAT - pin interrupt status.
34	GPIO_INT0_IRQ6	Pin interrupt 6 or pattern match engine slice 6 int	PSTAT - pin interrupt status.
35	GPIO_INT0_IRQ7	Pin interrupt 7 or pattern match engine slice 7 int	PSTAT - pin interrupt status.
36	CTIMER2	Standard counter/timer CTIMER2	Match and capture interrupts.

Table 8. Connection of interrupt sources to the NVIC *…continued*

Interrupt	Name	Interrupt description	Flags
37	CTIMER4	Standard counter/timer CTIMER4	Match and capture interrupts.
38	OSEVTIMER	OSTIMER0	-
39	Reserved	-	-
40	Reserved	-	-
41	Reserved	-	-
42	SDIO	SD/MMC interrupt	SDIO interrupts.
43	Reserved	-	-
44	Reserved	-	-
45	Reserved	-	-
46	USB1_PHY	USB1_PHY interrupts.	USB1_PHY interrupts.
47	USB1	USB1 interrupt	USB1 interrupts.
48	USB1_NEEDCLK	USB1 activity	USB1 interrupts.
49	HYPERVISOR	Hypervisor facilities	HF interrupts.
50	SGPIO_INT0_IRQ0	Secure GPIO function is available on P0(0-31) and 2x Pin Interrupt outputs are available to NVIC.	SGPIO 0 interrupts.
51	SGPIO_INT0_IRQ1	Secure GPIO function is available on P0(0-31) and 2x Pin Interrupt outputs are available to NVIC.	SGPIO 1 interrupts.
52	PLU	Programmable Logic Unit.	PLU interrupts.
53	SEC_VIO, SECURE_VIOLATION, SEC_VIOLATION	Secure violation interrupt.	Secure violation interrupts.
54	HASH	HASH interrupt.	Hash interrupts.
55	CASPER	CASPER Crypto co-processor interrupt.	Casper interrupts.
56	PUF	PUF Controller Interrupt.	PUF interrupts.
57	PQ	Power quad.	Power quad interrupts.
58	SDMA1	Secure DMA (DMA1) controller.	Secure DMA interrupts.
59	HS_SPI	HS_SPI	HS_SPI

Remark: When more than 1 wake-up sources is enabled (for example: RTC and WAKEIP), then the logic is:

a) If wake- up by WAKEIO pins -> only WAKEUP_IRQ is raised.

b) If wake-up by RTC -> WAKEUP_IRQ and RTC_IRQ are both raised.

If WAKEIOCAUSE register is 0, then wakeup is not done by WAKEIO.

If WAKEUP_IRQ is enabled, then you need to know how to identify wakeup source.

WAKEUP_IRQ Handler

{ if(WAKEIOCAUSE!=0)

//woke up by WAKEIO

else

```
//woke up by other source

}
```

3.4 Register description

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

Table 9. Register overview: NVIC (base address = 0xe000e100)

Name	Access	Offset	Description	Reset value	Section
ISER0	R/W	0x000	Interrupt set enable register 0. This register allows enabling interrupts and reading back the interrupt enables for peripheral functions.	0	3.4.1
ISER1	R/W	0x004	Interrupt set enable register 1. See ISER0 description.	0	3.4.2
ICER0	R/W	0x080	Interrupt clear enable register 0. This register allows disabling interrupts and reading back the interrupt enables for peripheral functions.	0	3.4.3
ICER1	R/W	0x084	Interrupt clear enable register 1. See ISER0 description.	0	3.4.4
ISPR0	R/W	0x100	Interrupt set pending register 0. This register allows changing the interrupt state to pending and reading back the interrupt pending state for peripheral functions.	0	3.4.5
ISPR1	R/W	0x104	Interrupt set pending register 1. See ISPR0 description.	0	3.4.6
ICPR0	R/W	0x180	Interrupt clear pending register 0. This register allows changing the interrupt state to not pending and reading back the interrupt pending state for peripheral functions.	0	3.4.7
ICPR1	R/W	0x184	Interrupt clear pending register 1. See ICPR0 description.	0	3.4.8
IABR0	RO	0x200	Interrupt active bit register 0. This register allows reading the current interrupt active state for specific peripheral functions.	0	3.4.9
IABR1	RO	0x204	Interrupt active bit register 1. See IABR0 description.	0	3.4.10
IPR0	R/W	0x300	Interrupt priority register 0. This register contains the 3-bit priority fields for interrupts 0 to 3.	0	3.4.11
IPR1	R/W	0x304	Interrupt priority register 1. This register contains the 3-bit priority fields for interrupts 4 to 7.	0	3.4.12
IPR2	R/W	0x308	Interrupt priority register2. This register contains the 3-bit priority fields for interrupts 8 to 11.	0	3.4.13
IPR3	R/W	0x30C	Interrupt priority register 3. This register contains the 3-bit priority fields for interrupts 12 to 15.	0	3.4.14
IPR4	R/W	0x310	Interrupt priority register 4. This register contains the 3-bit priority fields for interrupts 16 to 19.	0	3.4.15
IPR5	R/W	0x314	Interrupt priority register 5. This register contains the 3-bit priority fields for interrupts 20 to 23.	0	3.4.16
IPR6	R/W	0x318	Interrupt priority register 6. This register contains the 3-bit priority fields for interrupts 24 to 27.	0	3.4.17
IPR7	R/W	0x31C	Interrupt priority register 7. This register contains the 3-bit priority fields for interrupts 28 to 31.	0	3.4.18
IPR8	R/W	0x320	Interrupt priority register 8. This register contains the 3-bit priority fields for interrupts 32 to 35.	0	3.4.19
IPR9	R/W	0x324	Interrupt priority register 9. This register contains the 3-bit priority fields for interrupts 36 to 39.	0	3.4.20
IPR10	R/W	0x328	Interrupt priority register 10. This register contains the 3-bit priority fields for interrupts 40 to 43.	0	3.4.21
IPR11	R/W	0x32C	Interrupt priority register 11. This register contains the 3-bit priority fields for interrupts 44 to 47.	0	3.4.22
IPR12	R/W	0x330	Interrupt priority register 12. This register contains the 3-bit priority fields for interrupts 48 to 51	0	3.4.23

Table 9. Register overview: NVIC (base address = 0xe000e100) *…continued*

Name	Access	Offset	Description	Reset value	Section
IPR13	R/W	0x334	Interrupt priority register13. This register contains the 3-bit priority fields for interrupts 52 to 55.	0	3.4.24
IPR14	R/W	0x338	Interrupt priority register14. This register contains the 3-bit priority fields for interrupts 56 to 60.	0	3.4.25
IPR15	R/W	0x33C	Interrupt priority register15. This register contains the 3-bit priority fields for interrupts 61 to 63.	0	3.4.25
STIR	WO	0xE00	Software trigger interrupt register, allows software to generate interrupts.	-	3.4.26

3.4.1 Interrupt set-enable register 0

The ISER0 register allows enabling the first 32 peripheral interrupts, or for reading the enabled state of those interrupts. The remaining interrupts are enabled via the ISER1 register, see Section 3.4.2 “Interrupt set-enable register 1”. Disabling interrupts is done through the ICER0 and ICER1 registers Section 3.4.3 “Interrupt clear enable register 0” and Section 3.4.4 “Interrupt clear enable register 1”.

Table 10. Interrupt set-enable register 0

Bit	Name	Value	Function
0	ISE_WDTBOD	[1]	Watchdog Timer, BOD interrupt enable.
1	ISE_SDMA0	[1]	SDMA0 interrupt enable.
2	ISE_GINT0	[1]	GPIO group 0 interrupt enable.
3	ISE_GINT1	[1]	GPIO group 1 interrupt enable.
4	ISE_PINT0	[1]	Pin interrupt / pattern match engine slice 0 interrupt enable.
5	ISE_PINT1	[1]	Pin interrupt / pattern match engine slice 1 interrupt enable.
6	ISE_PINT2	[1]	Pin interrupt / pattern match engine slice 2 interrupt enable.
7	ISE_PINT3	[1]	Pin interrupt / pattern match engine slice 3 interrupt enable.
8	ISE_UTICK	[1]	Micro-Tick Timer interrupt enable.
9	ISE_MRT	[1]	Multi-Rate Timer interrupt enable.
10	ISE_CTIMER0	[1]	Standard counter/timer CTIMER0 interrupt enable.
11	ISE_CTIMER1	[1]	Standard counter/timer CTIMER1 interrupt enable.
12	ISE_SCT	[1]	SCT interrupt enable.
13	ISE_CTIMER3	[1]	Standard counter/timer CTIMER3 interrupt enable.
14	ISE_FC0	[1]	Flexcomm Interface 0 interrupt enable.
15	ISE_FC1	[1]	Flexcomm Interface 1 interrupt enable.
16	ISE_FC2	[1]	Flexcomm Interface 2 interrupt enable.
17	ISE_FC3	[1]	Flexcomm Interface 3 interrupt enable.
18	ISE_FC4	[1]	Flexcomm Interface 4 interrupt enable.
19	ISE_FC5	[1]	Flexcomm Interface 5 interrupt enable.
20	ISE_FC6	[1]	Flexcomm Interface 6 interrupt enable.
21	ISE_FC7	[1]	Flexcomm Interface 7 interrupt enable.
22	ISE_ADC0	[1]	ADC0 interrupt enable.
23	Reserved	[1]	-
24	ISE_ACMP	[1]	ACOMP interrupt enable.

Table 10. Interrupt set-enable register 0 *…continued*

Bit	Name	Value	Function
25	Reserved	[1]	-
26	Reserved	[1]	-
27	ISE_USB0_NEEDCLK	[1]	USB0 activity interrupt enable.
28	ISE_USB0	[1]	USB0 device interrupt enable.
29	ISE_RTC	[1]	Real Time Clock (RTC) interrupt enable.
30	-	[1]	Reserved
31	MAILBOX	[1]	MAILBOX

[1] 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.

3.4.2 Interrupt set-enable register 1

The ISER1 register allows enabling the second group of peripheral interrupts, or for reading the enabled state of those interrupts. Disabling interrupts is done through the ICER0 and ICER1 registers Section 3.4.3 “Interrupt clear enable register 0” and Section 3.4.4 “Interrupt clear enable register 1”.

Table 11. Interrupt set-enable register 1

Bit	Name	Value	Function
0	ISE_PINT4	[1]	Pin interrupt / pattern match engine slice 4 interrupt enable.
1	ISE_PINT5	[1]	Pin interrupt / pattern match engine slice 5 interrupt enable.
2	ISE_PINT6	[1]	Pin interrupt / pattern match engine slice 6 interrupt enable.
3	ISE_PINT7	[1]	Pin interrupt / pattern match engine slice 7 interrupt enable.
4	ISE_CTIMER2	[1]	Standard counter/timer CTIMER2 interrupt enable.
5	ISE_CTIMER4	[1]	Standard counter/timer CTIMER4 interrupt enable.
6	ISE_OSEVTIMER	[1]	OSTIMER0 interrupt enable.
7	-	[1]	Reserved.
8	-	[1]	Reserved.
9	-	[1]	Reserved.
10	ISE_SDIO	[1]	SD/MMC interrupt enable.
11	-	[1]	Reserved.
12	-	[1]	Reserved.
13	-	[1]	Reserved.
14	ISE_USB1_PHY	[1]	USB1_PHY interrupt enable.
15	ISE_USB1	[1]	USB1 device interrupt enable.
16	ISE_USB1_NEEDCLK	[1]	USB1 Activity Interrupt enable.
17	ISE_HYPERVISOR	[1]	Hypervisor facilities interrupt enable.
18	ISE_SGPIO_INT0_IRQ0	[1]	Secure GPIO interrupt enable.
19	ISE_SGPIO_INT0_IRQ1	[1]	Secure GPIO interrupt enable.
20	ISE_PLU	[1]	Programmable Logic Unit interrupt enable.
21	ISE_SECURE_VIOLATION	[1]	Security Violations interrupt enable.
22	ISE_HASH_AES	[1]	HASH_AES interrupt enable.
23	ISE_ CASPER	[1]	CASPER interrupt enable.

Table 11. Interrupt set-enable register 1 *…continued*

Bit	Name	Value	Function
24	ISE_PUF	[1]	PUF interrupt enable.
25	ISE_ PQ	[1]	Power quad interrupt enable.
26	ISE_ SDMA1	[1]	Secure DMA (DMA1) interrupt enable.
27	ISE_HS_SPI	[1]	FC8 interrupt enable.

[1] 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.

3.4.3 Interrupt clear enable register 0

The ICER0 register allows disabling the first 32 peripheral interrupts, or for reading the enabled state of those interrupts. The remaining interrupts are disabled via the ICER1 register Section 3.4.4 “Interrupt clear enable register 1”. Enabling interrupts is done through the ISER0 and ISER1 registers Section 3.4.1 “Interrupt set-enable register 0”and Section 3.4.2 “Interrupt set-enable register 1”

Table 12. Interrupt clear-enable register 0

Bit	Name	Function
31:0	ICE_...	Peripheral interrupt disables. Bit numbers match ISER0 registers Table 10. Unused bits are reserved. 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.

3.4.4 Interrupt clear enable register 1

The ICER1 register allows disabling the second group of peripheral interrupts, or for reading the enabled state of those interrupts. Enabling interrupts is done through the ISER0 and ISER1 registers Section 3.4.1 and Section 3.4.2.

Table 13. Interrupt clear-enable register 1

Bit	Name	Function
31:0	ICE_...	Peripheral interrupt disables. Bit numbers match ISER1 registers Table 11. Unused bits are reserved. 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.

3.4.5 Interrupt set pending register 0

The ISPR0 register allows setting the pending state of the first 32 peripheral interrupts, or for reading the pending state of those interrupts. The remaining interrupts can have their pending state set via the ISPR1 register Section 3.4.6 “Interrupt set pending register 1”. Clearing the pending state of interrupts is done through the ICPR0 and ICPR1 registers Section 3.4.7 “Interrupt clear pending register 0” and Section 3.4.8 “Interrupt clear pending register 1”.

Table 14. Interrupt set-pending register 0

Bit	Name	Function
31:0	ISP_...	Peripheral interrupt pending set. Bit numbers match ISER0 registers Table 10. Unused bits are reserved. 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.

3.4.6 Interrupt set pending register 1

The ISPR1 register allows setting the pending state of the second group of peripheral interrupts, or for reading the pending state of those interrupts. Clearing the pending state of interrupts is done through the ICPR0 and ICPR1 registers, Section 3.4.7 “Interrupt clear pending register 0” and Section 3.4.8 “Interrupt clear pending register 1”.

Table 15. Interrupt set-pending register 1

Bit	Name	Function
31:0	ISP_...	Peripheral interrupt pending set. Bit numbers match ISER1 registers Table 11. Unused bits are reserved. 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

3.4.7 Interrupt clear pending register 0

The ICPR0 register allows clearing the pending state of the first 32 peripheral interrupts, or for reading the pending state of those interrupts. The remaining interrupts can have their pending state cleared via the ICPR1 register Section 3.4.8 “Interrupt clear pending register 1”. Setting the pending state of interrupts is done through the ISPR0 and ISPR1 registers Section 3.4.5 “Interrupt set pending register 0” and Section 3.4.6 “Interrupt set pending register 1”.

Table 16. Interrupt clear-pending register 0

Bit	Name	Function
31:0	ICP_...	Peripheral interrupt pending clear. Bit numbers match ISER0 registers Table 10. Unused bits are reserved. 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.

3.4.8 Interrupt clear pending register 1

The ICPR1 register allows clearing the pending state of the second group of peripheral interrupts, or for reading the pending state of those interrupts. Setting the pending state of interrupts is done through the ISPR0 and ISPR1 registers Section 3.4.5 “Interrupt set pending register 0” and Section 3.4.6 “Interrupt set pending register 1”.

Table 17. Interrupt clear-pending register 1

Bit	Name	Function
31:0	ICP_...	Peripheral interrupt pending clear. Bit numbers match ISER1 registers Table 11. Unused bits are reserved. 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.

3.4.9 Interrupt active bit register 0

The IABR0 register is a read-only register that allows reading the active state of the first 32 peripheral interrupts. Bits in IABR are set while the corresponding interrupt service routines are in progress. Additional interrupts can have their active state read via the IABR1 register Section 3.4.10 “Interrupt active bit register 1”.

Table 18. Interrupt active bit register 0

Bit	Name	Function
31:0	IAB_...	Peripheral interrupt active. Bit numbers match ISER0 registers Table 10. Unused bits are reserved. Read: 0 indicates that the interrupt is not active, 1 indicates that the interrupt is active

3.4.10 Interrupt active bit register 1

The IABR1 register is a read-only register that allows reading the active state of the second peripheral interrupts. Bits in IABR are set while the corresponding interrupt service routines are in progress.

Table 19. Interrupt clear-pending register 1

Bit	Name	Function
31:0	IAB_...	Peripheral interrupt active. Bit numbers match ISER1 registers Table 11. Unused bits are reserved. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending.

3.4.11 Interrupt priority register 0

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

Table 20. Interrupt priority register 0

Bit	Name	Function
4:0	-	Unused.
7:5	IP_ WDT BOD FLASH	WDT BOD FLASH controller interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused.
15:13	IP_SDMA0	SDMA0 interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused.
23:21	IP_GINT0	GPIO Group 0 interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused.
31:29	IP_GINT1	GPIO Group 1 interrupt priority. 0 = highest priority. 7 = lowest priority.

3.4.12 Interrupt priority register 1

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

Table 21. Interrupt priority register 1

Bit	Name	Function
4:0	-	Unused.
7:5	IP_PINT0	Pin interrupt / pattern match engine slice 0 priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused.
15:13	IP_PINT1	Pin interrupt / pattern match engine slice 1 priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused.
23:21	IP_PINT2	Pin interrupt / pattern match engine slice 2 priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused.
31:29	IP_PINT3	Pin interrupt / pattern match engine slice 3 priority. 0 = highest priority. 7 = lowest priority.

3.4.13 Interrupt priority register 2

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

Table 22. Interrupt priority register 2

Bit	Name	Function
4:0	-	Unused.
7:5	IP_UTICK	Micro-Tick Timer interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused.
15:13	IP_MRT	Multi-Rate Timer interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused.
23:21	IP_CTIMER0	Standard counter/timer CTIMER0 interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused.
31:29	IP_CTIMER1	Pin interrupt / pattern match engine slice 3 priority. 0 = highest priority. 7 = lowest priority.

3.4.14 Interrupt priority register 3

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

Table 23. Interrupt priority register 3

Bit	Name	Function
4:0	-	Unused.
7:5	IP_SCT	SCT interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused.
15:13	IP_CTIMER3	Standard counter/timer CTIMER0 interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused.
23:21	IP_FC0	Flexcomm Interface 0 interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused.
31:29	IP_FC1	Flexcomm Interface 1 interrupt priority. 0 = highest priority. 7 = lowest priority.

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

3.4.15 Interrupt priority register 4

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

Table 24. Interrupt priority register 4

Bit	Name	Function
4:0	-	Unused.
7:5	IP_FC2	Flexcomm Interface 2 interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused.
15:13	IP_FC3	Flexcomm Interface 3 interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused.
23:21	IP_FC4	Flexcomm Interface 4 interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused.
31:29	IP_FC5	Flexcomm Interface 5 interrupt priority. 0 = highest priority. 7 = lowest priority.

3.4.16 Interrupt priority register 5

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

Table 25. Interrupt priority register 5

Bit	Name	Function
4:0	-	Unused.
7:5	IP_FC6	Flexcomm Interface 6 interrupt priority. 0 = highest priority. 7 = lowest priority
12:8	-	Unused.
15:13	IP_FC7	Flexcomm Interface 7 interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused.

Table 25. Interrupt priority register 5 *…continued*

Bit	Name	Function
23:21	IP_ADC0	ADC 0 interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused.
31:29	-	Unused.

3.4.17 Interrupt priority register 6

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

Table 26. Interrupt priority register 6

Bit	Name	Function
4:0	-	Unused.
7:5	IP_ ACMP	Analog Comparator interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused.
15:13	-	Unused.
20:16	-	Unused.
23:21	-	Unused.
28:24	-	Unused.
31:29	IP_USB0_NEEDCLK	USB0 Activity interrupt priority. 0 = highest priority. 7 = lowest priority.

3.4.18 Interrupt priority register 7

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

Table 27. Interrupt priority register 7

Bit	Name	Function
4:0	-	Unused.
7:5	IP_USB0	USB0 interrupt enable. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused.
15:13	IP_RTC	Real Time clock (RTC) interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused.
23:21	-	Unused.
28:24	-	Unused.
31:29	IP_MAILBOX	Mailbox priority. 0 = highest priority. 7 = lowest priority.

3.4.19 Interrupt priority register 8

The IPR8 register controls the priority of the ninth group of four peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 28. Interrupt priority register 8

Bit	Name	Function
4:0	-	Unused.
7:5	IP_PINT4	Pin interrupt / pattern match engine slice 4 priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused.

Table 28. Interrupt priority register 8 *…continued*

Bit	Name	Function
15:13	IP_PINT5	Pin interrupt / pattern match engine slice 5 priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused.
23:21	IP_PINT6	Pin interrupt / pattern match engine slice 6 priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused.
31:29	IP_PINT7	Pin interrupt / pattern match engine slice 7 priority. 0 = highest priority. 7 = lowest priority.

3.4.20 Interrupt priority register 9

The IPR9 register controls the priority of the tenth group of four peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 29. Interrupt priority register 9

Bit	Name	Function
4:0	-	Unused.
7:5	IP_CTIMER2	Standard counter/timer CTIMER2 interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused.
15:13	IP_CTIMER4	Standard counter/timer CTIMER4 interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused.
23:21	IP_ OSEVTIMER0	OSTIMER0 interrupt priority. 0 = highest priority. 7 = lowest priority.
31:24	-	Unused.

3.4.21 Interrupt priority register 10

The IPR10 register controls the priority of the eleventh group of four peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 30. Interrupt priority register 10

Bit	Name	Function
4:0	-	Unused.
7:5	Reserved	-
12:8	-	Unused.
15:13	Reserved	-
20:16	-	Unused.
23:21	IP_SDIO	SDIO interface interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused.
31:29	Reserved	-

3.4.22 Interrupt priority register 11

The IPR11 register controls the priority of the twelfth group of four peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 31. Interrupt priority register 11

Bit	Name	Function
4:0	-	Unused.
7:5	Reserved	-
12:8	-	Unused.
15:13	Reserved	-
20:16	-	Unused.
23:21	IP_USB1_PHY	USB1_PHY interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused.
31:29	IP_USB1	USB1 interface interrupt priority. 0 = highest priority. 7 = lowest priority.

3.4.23 Interrupt priority register 12

The IPR12 register controls the priority of the thirteenth group of four peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 32. Interrupt priority register 12

Bit	Name	Function
4:0	-	Unused.
7:5	IP_USB1_NEEDCLK	High speed USB interface interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused.
15:13	IP_HYPERVISOR	Hypervisor interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused.
23:21	IP_SGPIO_INT0_IRQ0	SGIO 0 interface interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused.
31:29	IP_SGPIO_INT0_IRQ1	SGIO 1 interface interrupt priority. 0 = highest priority. 7 = lowest priority.

3.4.24 Interrupt priority register 13

The IPR13 register controls the priority of the fourteenth group of four peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 33. Interrupt priority register 13

Bit	Name	Function
4:0	-	Unused.
7:5	IP_PLU	Programmable Logic Unit interface interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused.
15:13	IP_SECURE_VIOLATION	Secure Violation interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused.
23:21	IP_HASH_AES	HASH_AES interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused.
31:29	IP_CASPER	Casper interface interrupt priority. 0 = highest priority. 7 = lowest priority.

3.4.25 Interrupt priority register 14

The IPR14 register controls the priority of the fifteenth group of four peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 34. Interrupt priority register 14

Bit	Name	Function
4:0	-	Unused.
7:5	IP_PUF_IRQ	PUF interface interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused.
15:13	IP_PQ	Power quad interface interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused.
23:21	IP_SDMA1	Secure DMA interface interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused.
31:29	IP_ HS_SPI	FC8 or HS_SPI interface interrupt priority. 0 = highest priority. 7 = lowest priority.

3.4.26 Software trigger interrupt register

The STIR register provides an alternate way for software to generate an interrupt, in addition to using the ISPR registers. This mechanism can only be used to generate peripheral interrupts, not system exceptions.
By default, only privileged software can write to the STIR register. Unprivileged software can be given this ability if privileged software sets the USERSETMPEND bit in the CCR register.

The interrupt number to be programmed in this register is listed in Table 35.

Table 35. Software trigger interrupt register (STIR)

Bit	Name	Function
8:0	INTID	Writing a value to this field generates an interrupt for the specified the interrupt number.
31:9	-	Reserved. Read value is undefined, only zero should be written.

4.1 Features

- System and bus configuration.
- Clock select and control.
- PLL0 and PLL1 configuration.
- Reset control.
- Wake-up control.
- High-accuracy frequency measurement function for on-chip and off-chip clocks.
- Uses a selection of on-chip clocks as reference clock.
- Device ID register.

4.2 Basic configuration

Configure the SYSCON block as follows:

- No clock configuration is needed. The clock to the SYSCON block is always enabled. By default, the SYSCON block is clocked by the FRO 12 MHz (fro_12m).
- Target and reference clocks for the frequency measurement function are selected in the input mux block. See Chapter 18 “LPC55S6x/LPC55S2x/LPC552x Input Multiplexing (INPUTMUX)”.
- The SYSCON block controls use of the CLKOUT pin, which must also be configured through IOCON. See Section 4.3 “Pin description”. RESET is a dedicated pin.

4.2.1 Set up the PLL0

The PLL0 creates a stable output clock at a higher frequency than the input clock. If a main clock is needed with a frequency higher than the FRO 12 MHz clock and the FRO 96 MHz clock (fro_hf) is not appropriate, use the PLL to boost the input frequency.

4.2.2 Set up the PLL1

The PLL1 creates a stable output clock at a higher frequency than the input clock. If a main clock is needed with a frequency higher than the FRO 12 MHz clock and the FRO 96 MHz clock (fro_hf) is not appropriate, use the PLL to boost the input frequency.

4.2.3 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 selected from the sources listed in step 1 below.

The main clock, after being optionally divided by the CPU Clock Divider, is called the system clock and clocks the core, the memories, and the peripherals (register interfaces and peripheral clocks).

1. Select the main clock. The following options are available:

- FRO 12 MHz output (fro_12m) from internal oscillator (default). This clock is divided down from FRO high-speed.
- FRO high speed output (fro_hf), 96 MHz from internal oscillator.
- External oscillator.
- FRO 1 MHz output (fro_1m) from internal oscillator.
- The output of the PLL0.
- The output of the PLL1.
- The RTC 32 kHz oscillator.

Section 4.5.34 “Main clock source select register A” and Section 4.5.35 “Main clock source select register B”.

2. Select the divider value for the system clock Section 4.5.50 “AHB clock divider register”.
3. Enable the clock to the memories and peripherals used in the application.

4.2.4 Measure the frequency of a clock signal

The frequency of any on-chip or off-chip clock signal can be measured accurately with a selectable reference clock. For example, the frequency measurement function can be used to accurately determine the frequency of the watchdog oscillator which varies over a wide range depending on process and temperature.

The clock frequency to be measured and the reference clock are selected in the input mux block. See Section 18.6.9 “Frequency measure function reference clock select register”, Section 11.5.3 “Frequency measure function control register ”, and Section 18.6.10 “Frequency measure function target clock select register”.

Details on the accuracy and measurement process are described in Section 11.6.1 “Frequency measure function”.

To start a frequency measurement cycle and read the result, see

4.3 Pin description

Table 36. SYSCON pin description

Function	Type	Pin	Description	Reference
CLKOUT	O	PIO0_16, PIO0_26, PIO1_27	Clock output	Chapter 15 “LPC55S6x/LPC55S2x/LPC552x I/O Pin Configuration (IOCON)”

4.4 General description

4.4.1 Clock generation

The system control block facilitates the clock generation. Many clocking variations are possible. Figure 4 “LPC55S6x Clock generation (Part 1 of 3)” gives an overview of potential clock options. Table 37 “Clocking diagram signal name descriptions” describes signals on the clocking diagram. Device revision 1B operates at a maximum CPU frequency of up to 150 MHz. Device revision 0A operates at a maximum CPU frequency of up to 100 MHz.

Remark: The indicated clock multiplexers shown in Figure 4 “LPC55S6x Clock generation (Part 1 of 3)” are synchronized. In order to operate, the currently selected clock must be running, and the clock to be switched to must also be running so the multiplexer can gracefully switch between the two clocks without glitches. Other clock multiplexers are not synchronized. The output divider can be stopped and restarted gracefully during switching if a glitch-free output is needed.

The low-power oscillator provides a frequency in the range of 1 MHz. The accuracy of this clock is limited to +/- 15% over temperature, voltage, and silicon processing variations after trimming made during assembly. To determine the actual watchdog oscillator output, use the frequency measure block. See Section 4.2.4 “Measure the frequency of a clock signal”.

The device contains two PLLs (PLL0 and PLL1) that can be configured to use a number of clock inputs and produce an output clock in the range of 1.2 MHz up to the maximum chip frequency, and can be used to run most on-chip functions. The output of the PLL can be monitored through the CLKOUT pin.

Remark: The maximum allowed frequency for main clock and system clock (to CPU0, CPU1, AHB bus, Sync, etc.) is 150 MHz. See Figure 4 “LPC55S6x Clock generation (Part 1 of 3)”. The POWER_SetVoltageForFreq API call must always be used when setting or switching the frequency. See Chapter 14 “LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API”.

Table 37. Clocking diagram signal name descriptions

Name	Description
32k_osc	The 32 kHz clock source. It is selected as either FRO32K or XTAL32K in the RTCOSCCTRL register.
clk_in	It is the internal clock that comes from the external oscillator.
frg_clk	The output of each Fractional Rate Generator to Flexcomm clock. Each FRG and its source selection is shown in Figure 4.
fro_12m	12 MHz divided down from the currently selected on-chip FRO_192 oscillator.
fro_hf	The currently selected FRO_192 high speed output at 96 MHz. FRO_HF clock is the output of the FRO_192 divided by 2 (96 MHz). Note that this clock can only be used for USB device and is not reliable for USB host timing requirements of the data signaling rate.
main_clk	The main clock used by the CPU and AHB bus, and potentially many others. The main clock and its source selection are shown in Figure 4.
mclk_in	The MCLK input function, when it is connected to a pin by selecting it in the IOCON block.
pll0_clk	The output of the PLL0. The PLL0 and its source selection is shown in Figure 4.

Table 37. Clocking diagram signal name descriptions *...continued*

Name	Description
pll1_clk	The output of the PLL1. The PLL1 and its source selection is shown in Figure 4.
fro_1m	The output of the low power oscillator.
“none”	A tied-off source that should be selected to save power when the output of the related multiplexer is not used.

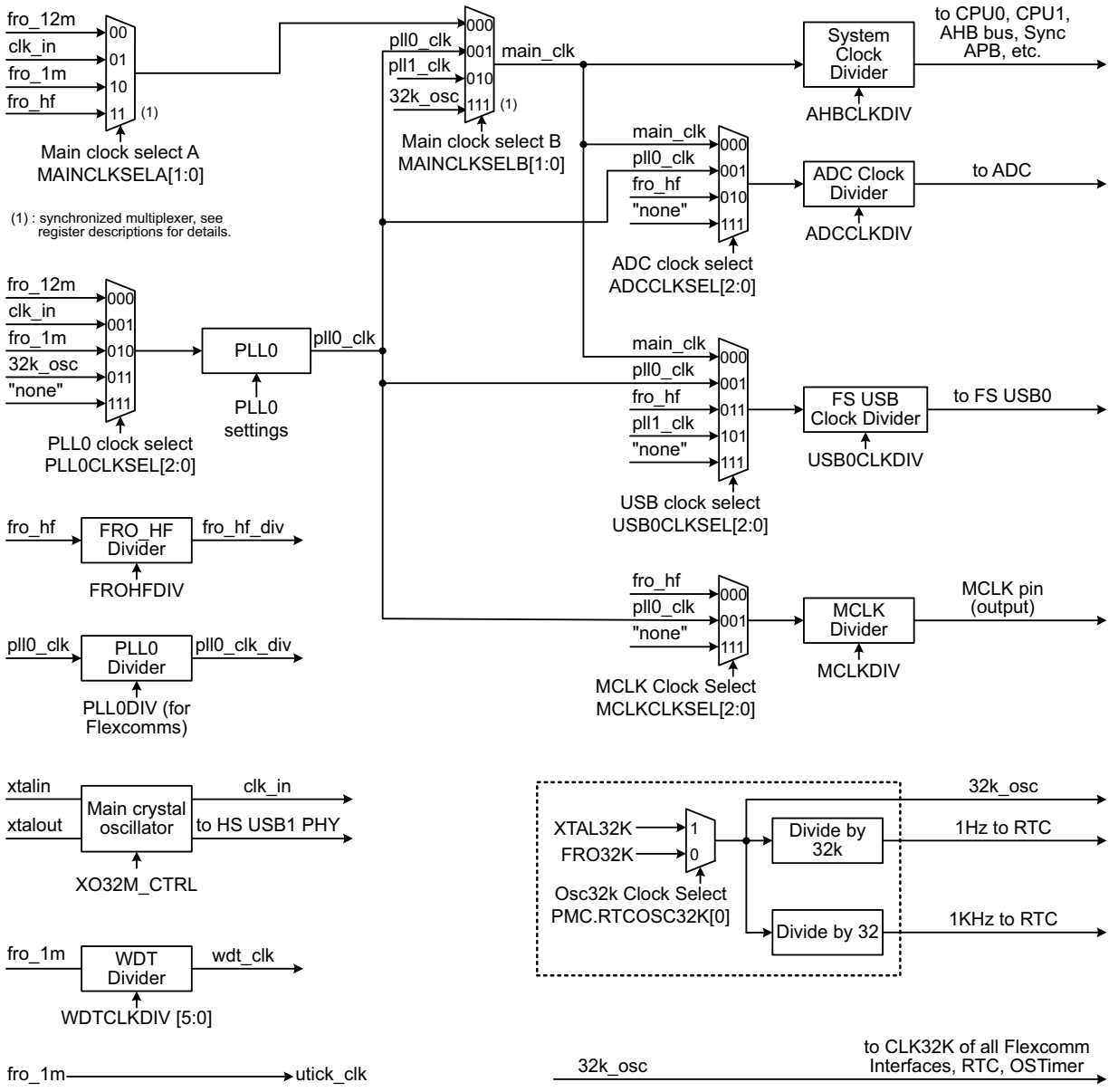


Fig 4. LPC55S6x Clock generation (Part 1 of 3)

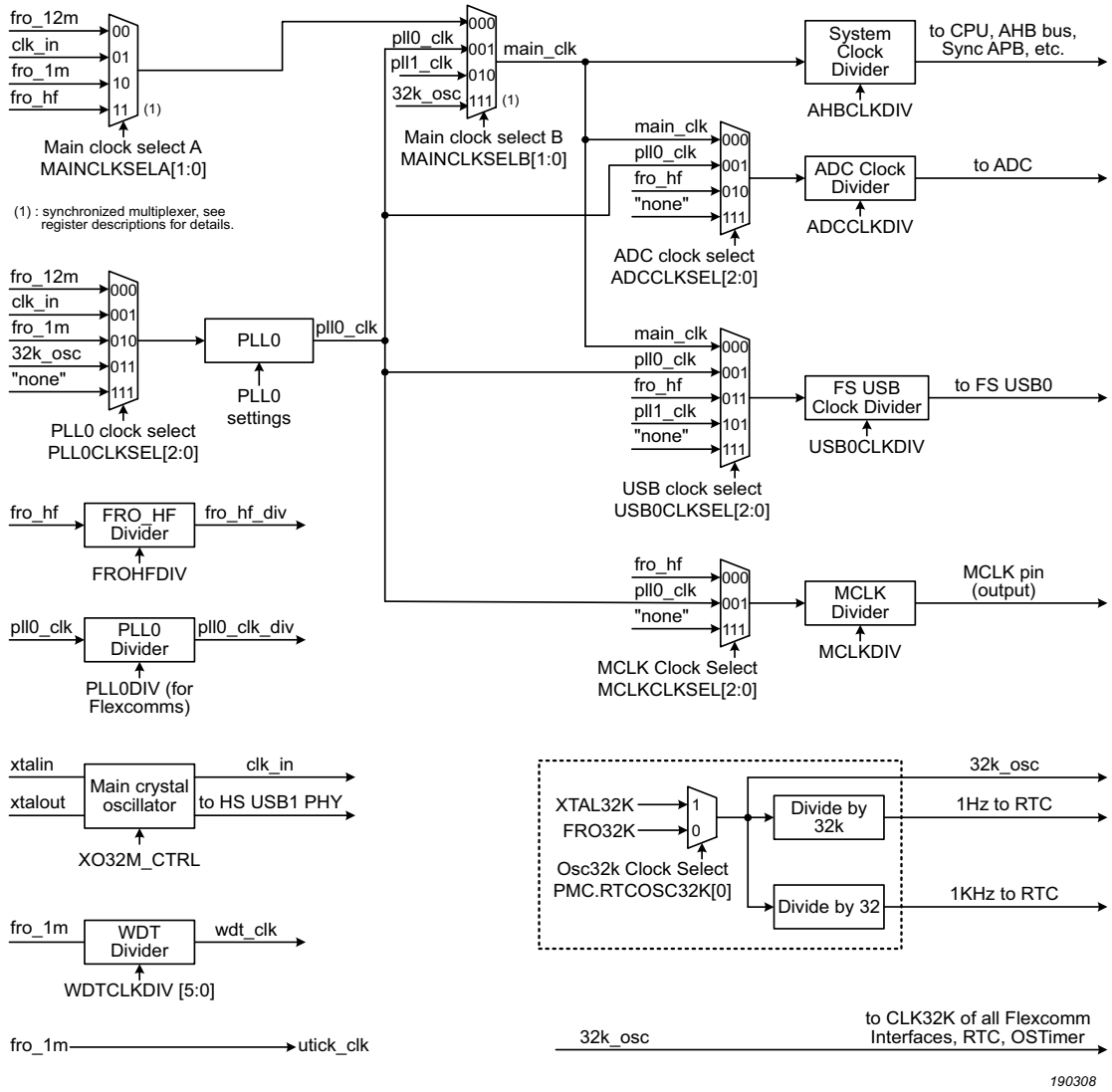


Fig 5. Clock generation LPC552x/LPC55S2x (Part 2 of 3)

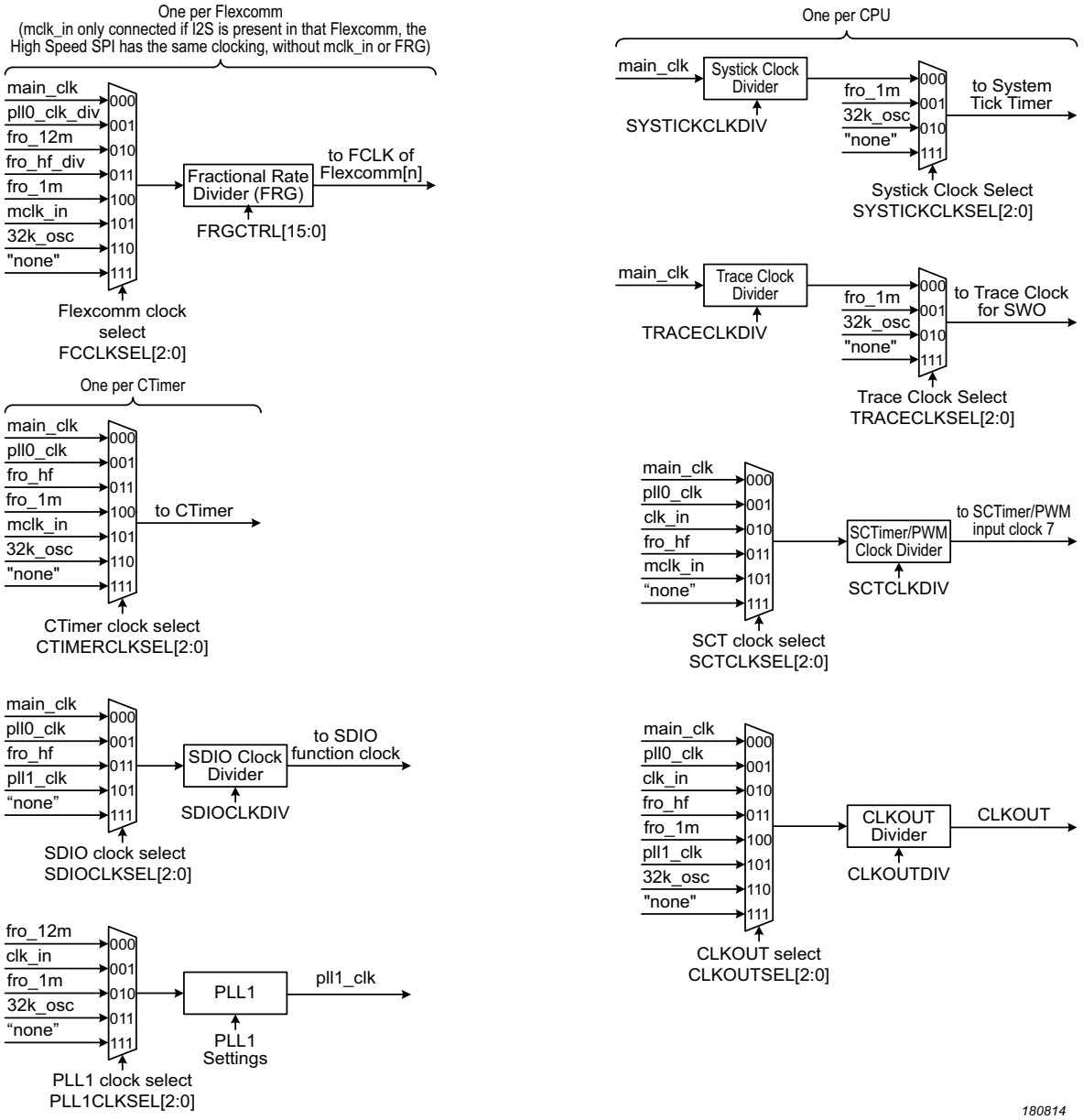


Fig 6. LPC55S6x/LPC55S2x/LPC552x Clock generation (Part 3 of 3)

4.5 Register description

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

- Main system configuration at base address 0x5000 0000,see Table 38 is secure and 0x4000 0000 is non-secure.

All address offsets not shown in the tables are reserved and should not be written to.

Remark: The reset value column shows the reset value seen when the boot loader executes and the flash contains valid user code. During code development, a different value may be seen if a debugger is used to halt execution prior to boot completion.

Table 38. Register overview: SYSCON (base address = 0x50000000)

Name	Access	Offset	Description	Reset value	Section
MEMORYREMAP	RW	0x0	Memory remap control register.	0x0	4.5.1
AHBMATPRIO	RW	0x10	AHB Matrix priority control register priority values are 3 = highest, 0 = lowest	0x0	4.5.2
CPU0STCKCAL	RW	0x38	System tick calibration for secure part of CPU0.	0x0	4.5.3
CPU0NSTCKCAL	RW	0x3C	System tick calibration for non-secure part of CPU0.	0x0	4.5.4
CPU1TCKCAL	RW	0x40	System tick calibration for CPU1.	0x0	4.5.5
NMISRC	RW	0x48	NMI source select.	0x0	4.5.6
PRESETCTRL0	RW	0x100	Peripheral reset control 0.	0x0	4.5.7
PRESETCTRL1	RW	0x104	Peripheral reset control 1.	0x0	4.5.8
PRESETCTRL2	RW	0x108	Peripheral reset control 2.	0x0	4.5.9
PRESETCTRLSET0	RW	0x120	Peripheral reset control set register.	0x0	4.5.10
PRESETCTRLSET1	RW	0x124	Peripheral reset control set register.	0x0	4.5.11
PRESETCTRLSET2	RW	0x128	Peripheral reset control set register.	0x0	4.5.12
PRESETCTRLCLR0	RW	0x140	Peripheral reset control clear register.	0x0	4.5.13
PRESETCTRLCLR1	RW	0x144	Peripheral reset control clear register.	0x0	4.5.14
PRESETCTRLCLR2	RW	0x148	Peripheral reset control clear register.	0x0	4.5.15
SWR_RESET	W	0x160	Generate a software reset.	0x0	4.5.16
AHBCLKCTRL0	RW	0x200	AHB clock control 0.	0x180	4.5.17
AHBCLKCTRL1	RW	0x204	AHB clock control 1.	0x0	4.5.18
AHBCLKCTRL2	RW	0x208	AHB clock control 2.	0x0	4.5.19
AHBCLKCTRLSET0	RW	0x220	Peripheral reset control register.	0x0	4.5.20
AHBCLKCTRLSET1	RW	0x224	Peripheral reset control register.	0x0	4.5.21
AHBCLKCTRLSET2	RW	0x228	Peripheral reset control register.	0x0	4.5.22
AHBCLKCTRLCLR0	RW	0x240	Peripheral reset control register.	0x0	4.5.23
AHBCLKCTRLCLR1	RW	0x244	Peripheral reset control register.	0x0	4.5.24
AHBCLKCTRLCLR2	RW	0x248	Peripheral reset control register.	0x0	4.5.25
SYSTICKCLKSEL0	RW	0x260	System Tick Timer for CPU0 source select.	0x0	4.5.26
SYSTICKCLKSEL1	RW	0x264	System Tick Timer for CPU1 source select.	0x0	4.5.27
TRACECLKSEL	RW	0x268	Trace clock source select.	0x0	4.5.28

Table 38. Register overview: SYSCON (base address = 0x50000000) *…continued*

Name	Access	Offset	Description	Reset value	Section
CTIMERCLKSEL0	RW	0x26C	CTimer 0 clock source select.	0x0	4.5.29
CTIMERCLKSEL1	RW	0x270	CTimer 1 clock source select.	0x0	4.5.30
CTIMERCLKSEL2	RW	0x274	CTimer 2 clock source select.	0x0	4.5.31
CTIMERCLKSEL3	RW	0x278	CTimer 3 clock source select.	0x0	4.5.32
CTIMERCLKSEL4	RW	0x27C	CTimer 4 clock source select.	0x0	4.5.33
MAINCLKSELA	RW	0x280	Main clock A source select.	0x0	4.5.34
MAINCLKSELB	RW	0x284	Main clock B source select.	0x0	4.5.35
CLKOUTSEL	RW	0x288	CLKOUT clock source select.	0x7	4.5.36
PLL0CLKSEL	RW	0x290	PLL0 clock source select.	0x7	4.5.37
PLL1CLKSEL	RW	0x294	PLL1 clock source select.	0x7	4.5.38
ADCCLKSEL	RW	0x2A4	ADC clock source select.	0x7	4.5.39
USB0CLKSEL	RW	0x2A8	FS USB clock source select.	0x7	4.5.40
FCCLKSEL0	RW	0x2B0	Flexcomm Interface 0 clock source select for Fractional Rate Divider.	0x7	4.5.41
FCCLKSEL1	RW	0x2B4	Flexcomm Interface 1 clock source select for Fractional Rate Divider.	0x7	4.5.41
FCCLKSEL2	RW	0x2B8	Flexcomm Interface 2 clock source select for Fractional Rate Divider.	0x7	4.5.41
FCCLKSEL3	RW	0x2BC	Flexcomm Interface 3 clock source select for Fractional Rate Divider.	0x7	4.5.41
FCCLKSEL4	RW	0x2C0	Flexcomm Interface 4 clock source select for Fractional Rate Divider.	0x7	4.5.41
FCCLKSEL5	RW	0x2C4	Flexcomm Interface 5 clock source select for Fractional Rate Divider.	0x7	4.5.41
FCCLKSEL6	RW	0x2C8	Flexcomm Interface 6 clock source select for Fractional Rate Divider.	0x7	4.5.41
FCCLKSEL7	RW	0x2CC	Flexcomm Interface 7 clock source select for Fractional Rate Divider.	0x7	4.5.41
HSLSPICLKSEL	RW	0x2D0	HS SPI clock source select.	0x7	4.5.42
MCLKCLKSEL	RW	0x2E0	I^2S MCLK clock source select.	0x7	4.5.43
SCTCLKSEL	RW	0x2F0	SCTimer/PWM clock source select.	0x7	4.5.44
SDIOCLKSEL	RW	0x2F8	SDIO clock source select.	0x7	4.5.45
SYSTICKCLKDIV0	RW	0x300	System Tick Timer divider for CPU0.	0x4000000	4.5.46
SYSTICKCLKDIV1	RW	0x304	System Tick Timer divider for CPU1.	0x4000000	4.5.47
TRACECLKDIV	RW	0x308	TRACE clock divider.	0x4000000	4.5.48
FLEXFRG0CTRL	RW	0x320	Fractional Rate Divider for Flexcomm Interface 0.	0x0	4.5.49
FLEXFRG1CTRL	RW	0x324	Fractional Rate Divider for Flexcomm Interface 1.	0x0	4.5.49
FLEXFRG2CTRL	RW	0x328	Fractional Rate Divider for Flexcomm Interface 2.	0x0	4.5.49
FLEXFRG3CTRL	RW	0x32C	Fractional Rate Divider for Flexcomm Interface 3.	0x0	4.5.49

Table 38. Register overview: SYSCON (base address = 0x50000000) *…continued*

Name	Access	Offset	Description	Reset value	Section
FLEXFRG4CTRL	RW	0x330	Fractional Rate Divider for Flexcomm Interface 4.	0x0	4.5.49
FLEXFRG5CTRL	RW	0x334	Fractional Rate Divider for Flexcomm Interface 5.	0x0	4.5.49
FLEXFRG6CTRL	RW	0x338	Fractional Rate Divider for Flexcomm Interface 6.	0x0	4.5.49
FLEXFRG7CTRL	RW	0x33C	Fractional Rate Divider for Flexcomm Interface 7.	0x0	4.5.49
AHBCLKDIV	RW	0x380	System clock divider.	0x0	4.5.50
CLKOUTDIV	RW	0x384	CLKOUT clock divider.	0x4000000	4.5.51
FROHFDIV	RW	0x388	FRO_HF. FRO_HF clock is the output of the FRO_192 divided by 2 (96 MHz).	0x4000000	4.5.52
WDTCLKDIV	RW	0x38C	WDT clock divider.	0x4000000	4.5.53
ADCCLKDIV	RW	0x394	ADC clock divider.	0x4000000	4.5.54
USB0CLKDIV	RW	0x398	USB0 Clock divider.	0x4000000	4.5.55
MCLKDIV	RW	0x3AC	I^2S MCLK clock divider.	0x4000000	4.5.56
SCTCLKDIV	RW	0x3B4	SCT/PWM clock divider.	0x4000000	4.5.57
SDIOCLKDIV	RW	0x3BC	SDIO clock divider.	0x4000000	4.5.58
PLL0CLKDIV	RW	0x3C4	PLL0 clock divider.	0x4000000	4.5.59
CLOCKGENUPDATELOCKOUT	RW	0x3FC	Control clock configuration registers access.	0x0	4.5.60
FMCCR	RW	0x400	FMC configuration register.	0x0	4.5.61
USB0NEEDCLKCTRL	RW	0x40C	USB0 need clock control.	0x0	4.5.62
USB0NEEDCLKSTAT	R	0x410	USB0 need clock status.	0x0	4.5.63
FMCFLUSH	W	0x41C	FMC flush control.	0x0	4.5.64
MCLKIO	RW	0x420	MCLK control.	0x0	4.5.65
USB1NEEDCLKCTRL	RW	0x424	USB1 need clock control.	0x10	4.5.66
USB1NEEDCLKSTAT	RW	0x428	USB1 need clock status.	0x0	4.5.67
SDIOCLKCTRL	W	0x460	SDIO CCLKIN phase and delay control.	0x0	4.5.68
PLL1CTRL	RW	0x560	PLL1 control.	0x0	4.5.69
PLL1STAT	R	0x564	PLL1 status.	0x0	4.5.69
PLL1NDEC	RW	0x568	PLL1 N divider.	0x0	4.5.69
PLL1MDEC	RW	0x56C	PLL1 M divider.	0x0	4.5.69
PLL1PDEC	RW	0x570	PLL1 P divider.	0x0	4.5.69
PLL0CTRL	RW	0x580	PLL0 control.	0x0	4.5.69
PLL0STAT	R	0x584	PLL0 status.	0x0	4.5.69
PLL0NDEC	RW	0x588	PLL0 N divider.	0x0	4.5.69
PLL0PDEC	RW	0x58C	PLL0 P divider.	0x0	4.5.69
PLL0SSCG0	RW	0x590	System PLL Spread Spectrum Wrapper control register 0.	0x0	4.5.69
PLL0SSCG1	RW	0x594	System PLL Spread Spectrum Wrapper control register 1.	0x0	4.5.69
FUNCRETENTIONCTRL	RW	0x704	Functional retention control register.	0x50C000	4.5.70
CPUCTRL	RW	0x800	CPU Control for multiple processors.	0x0	4.5.71

Table 38. Register overview: SYSCON (base address = 0x50000000) *…continued*

Name	Access	Offset	Description	Reset value	Section
CPBOOT	RW	0x804	Coprocessor boot address.	0x0	4.5.72
CPSTAT	RW	0x80C	CPU Status.	0x0	4.5.73
DICE_REG0	RW	0x900	Can be used as scratch register.	0x0	4.5.74
DICE_REG1	RW	0x904	Can be used as scratch register.	0x0	4.5.74
DICE_REG2	RW	0x908	Can be used as scratch register.	0x0	4.5.74
DICE_REG3	RW	0x90C	Can be used as scratch register.	0x0	4.5.74
DICE_REG4	RW	0x910	Can be used as scratch register.	0x0	4.5.74
DICE_REG5	RW	0x914	Can be used as scratch register.	0x0	4.5.74
DICE_REG6	RW	0x918	Can be used as scratch register.	0x0	4.5.74
DICE_REG7	RW	0x91C	Can be used as scratch register.	0x0	4.5.74
CLOCK_CTRL	RW	0xA18	Various system clock controls: Flash clock (48 MHz) control, clocks to Frequency Measures.	0x0	4.5.75
COMP_INT_CTRL	RW	0xB10	Comparator interrupt control.	0x0	4.5.76
COMP_INT_STATUS	R	0xB14	Comparator interrupt status.	0x0	4.5.77
AUTOCLKGATEOVERRIDE	RW	0xE04	Control automatic clock gating.	0xFFFF	4.5.78
GPIOPSYNC	RW	0xE08	Enable bypass of the first stage of synchronization inside GPIO_INT module.	0x0	4.5.79
DEBUG_LOCK_EN	RW	0xFA0	Control write access to security registers.	0x0	4.5.80
DEBUG_FEATURES	RW	0xFA4	CPU0 and CPU1 debug features control.	0x0	4.5.81
DEBUG_FEATURES_DP	RW	0xFA8	CPU0 and CPU1 debug features control DUPLICATE register.	0x0	4.5.82
SWD_ACCESS_CPU0	RW	0xFB4	Enable SWD debug access for CPU0.	0x0	4.5.83
SWD_ACCESS_CPU1	RW	0xFB8	Enable SWD debug access for CPU1.	0x0	4.5.84
KEY_BLOCK	W	0xFBC	Block access to PUF indexes.	0x0	4.5.85
DEBUG_AUTH_BEACON	RW	0xFC0	Debug authentication BEACON register.	0x0	4.5.86
CPUCFG	RW	0xFD4	CPU’s configuration register.	0x0	4.5.87
DEVICE_ID0	R	0xFF8	Device ID register.	0x0	4.5.88
DIEID	R	0xFFC	Chip revision ID and number.	0x0	4.5.89

4.5.1 Memory remap control register

The memory remap control selects the memory location of the vector table.

Table 39. Memory remap control register (MEMORYREMAP, offset = 0x0)

Bit	Symbol	Value	Description	Reset value
1:0	MAP		Select the location of the vector table.	0x0
		0	Vector table in ROM.	
		1	Vector table in RAM.	
		2	Vector table in flash.	
		3	Vector table in flash.	
31:2	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.2 AHB matrix priority register

The multilayer AHB matrix arbitrates between several masters, only if they attempt to access the same matrix slave port at the same time. Care should be taken if the value in this register is changed. Improper settings can seriously degrade performance.

Priority values are 3 = highest, 0 = lowest. When the priority is the same, the master with the lower master number is given priority.

Table 40. AHB Matrix priority control register (AHBMATPRIO, offset = 0x10)

Bit	Symbol	Value	Description	Reset value
1:0	CPU0 Code-bus		CPU0 C-AHB bus.	0x0
3:2	CPU0 System-bus		CPU0 S-AHB bus.	0x0
5:4	CPU1 Code-bus		CPU1 C-AHB bus.	0x0
7:6	CPU1 System-bus		CPU1 S-AHB bus.	0x0
9:8	USB-FS Device		USB0-FS (USB0).	0x0
11:10	SDMA0		SDMA0 controller priority.	0x0
13:12	-		Reserved. Read value is undefined, only zero should be written.	undefined
15:14	-		Reserved. Read value is undefined, only zero should be written.	undefined
17:16	SDIO		SDIO.	0x0
19:18	PowerQuad (DSP HW Accelerator)		PQ (HW accelerator).	0x0
21:20	SHA-2		HASH_AES	0x0
23:22	USB-FS Host		USB-FS (USB1).	0x0
25:24	SDMA1 (Secure)		SDMA1 controller priority.	0x0
31:26	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.3 System tick calibration for secure part of CPU0

This register allows software to set up a default value for the SYST_CALIB register in the System Tick Timer of secure part of the CPU0. See Chapter 29 “LPC55S6x/LPC55S2x/LPC552x System Tick Timer”

Table 41. System tick calibration for secure part of CPU0 (CPU0STCKCAL, offset = 0x38)

Bit	Symbol	Description	Reset value
23:0	TENMS	Reload value for 10ms (100Hz) timing, subject to system clock skew errors. If the value reads as zero, the calibration value is not known.	0x0
24	SKEW	Indicates whether the TENMS value is exact: 0 = TENMS value is exact; 1 = TENMS value is inexact, or not given.	0x0
25	NOREF	Indicates whether the device provides a reference clock to the processor: 0 = reference clock provided; 1 = no reference clock provided.	0x0
31:26	-	Reserved. Read value is undefined, only zero should be written.	undefined

4.5.4 System tick calibration for non-secure part of CPU0

This register allows software to set up a default value for the SYST_CALIB register in the System Tick Timer of non-secure part of the CPU0. See Chapter 29 “LPC55S6x/LPC55S2x/LPC552x System Tick Timer”

Table 42. System tick calibration for non-secure part of CPU0 (CPU0NSTCKCAL, offset = 0x3C)

Bit	Symbol	Description	Reset value
23:0	TENMS	Reload value for 10ms (100Hz) timing, subject to system clock skew errors. If the value reads as zero, the calibration value is not known.	0x0
24	SKEW	Indicates whether the TENMS value is exact: 0 = TENMS value is exact; 1 = TENMS value is inexact, or not given.	0x0
25	NOREF	Indicates whether the device provides a reference clock to the processor: 0 = reference clock provided; 1 = no reference clock provided.	0x0
31:26	-	Reserved. Read value is undefined, only zero should be written.	undefined

4.5.5 System tick calibration for CPU1

This register allows software to set up a default value for the SYST_CALIB register in the System Tick Timer of CPU1. See Chapter 29 “LPC55S6x/LPC55S2x/LPC552x System Tick Timer”

Table 43. System tick calibration for CPU1 (CPU1STCKCAL, offset = 0x40)

Bit	Symbol	Description	Reset value
23:0	TENMS	Reload value for 10ms (100Hz) timing, subject to system clock skew errors. If the value reads as zero, the calibration value is not known.	0x0
24	SKEW	Indicates whether the TENMS value is exact: 0 = TENMS value is exact; 1 = TENMS value is inexact, or not given.	0x0
25	NOREF	Indicates whether the device provides a reference clock to the processor: 0 = reference clock provided; 1 = no reference clock provided.	0x0
31:26		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.6 NMI source selection register

The NMI source selection register selects a peripheral interrupts as source for the NMI interrupt of both CPUs. For a list of all peripheral interrupts and their IRQ numbers, see Table 44. For a description of the NMI functionality, Ref. 1 “Cortex-M33 DGUG”

Remark: To change the interrupt source for the NMI, the NMI source must first be disabled by writing 0 to the NMIEN bit. Then change the source by updating the IRQN bits and re-enabling the NMI source by setting NMIEN.

Table 44. NMI source select (NMISRC, offset = 0x48)

Bit	Symbol	Description	Reset value
5:0	IRQCPU0	The IRQ number of the interrupt that acts as the Non-Maskable Interrupt (NMI) for the CPU0, if enabled by NMIENCPU0.	0x0
7:6	-	Reserved. Read value is undefined, only zero should be written.	undefined
13:8	IRQCPU1	The IRQ number of the interrupt that acts as the Non-Maskable Interrupt (NMI) for the CPU1, if enabled by NMIENCPU1.	0x0
29:14	-	Reserved. Read value is undefined, only zero should be written.	undefined
30	NMIENCPU1	Write a 1 to this bit to enable the Non-Maskable Interrupt (NMI) source selected by IRQCPU1.	0x0
31	NMIENCPU0	Write a 1 to this bit to enable the Non-Maskable Interrupt (NMI) source selected by IRQCPU0.	0x0

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. It can be avoided by disabling the normal interrupt in the NVIC.

4.5.7 Peripheral reset control 0

The PRESETCTRL0 register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a 1 asserts the reset.

Remark: It is recommended that changes to the PRESETCTRL registers be accomplished by using the related PRESETCTRLSET and PRESETCTRLCLR registers. It avoids any unintentional setting or clearing of other bits.

Table 45. Peripheral reset control 0 (PRESETCTRL0, offset = 0x100)

Bit	Symbol	Value	Description	Reset value
0	-		Reserved. Read value is undefined, only zero should be written.	undefined
1	ROM_RST		ROM reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
2	-		Reserved. Read value is undefined, only zero should be written.	undefined
3	SRAM_CTRL1_RST		SRAM controller 1 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
4	SRAM_CTRL2_RST		SRAM controller 2 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
5	SRAM_CTRL3_RST		SRAM controller 3 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
6	SRAM_CTRL4_RST		SRAM controller 4 reset control	0x0
		1	Block is reset.	
		0	Block is not reset.	

Table 45. Peripheral reset control 0 (PRESETCTRL0, offset = 0x100) *…continued*

Bit	Symbol	Value	Description	Reset value
7	FLASH_RST		Flash controller reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
8	FMC_RST		FMC controller reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
10:9	-		Reserved. Read value is undefined, only zero should be written.	undefined
11	MUX_RST		Input MUX reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
12	-		Reserved. Read value is undefined, only zero should be written.	undefined
13	IOCON_RST		I/O controller reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
14	GPIO0_RST		GPIO0 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
15	GPIO1_RST		GPIO1 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
17:16	-		Reserved. Read value is undefined, only zero should be written.	undefined
18	PINT_RST		Pin interrupt (PINT) reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
19	GINT_RST		Group interrupt (PINT) reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
20	DMA0_RST		DMA0 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
21	CRCGEN_RST		CRCGEN reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
22	WWDT_RST		Watchdog Timer reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
23	RTC_RST		Real Time Clock (RTC) reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
25:24	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 45. Peripheral reset control 0 (PRESETCTRL0, offset = 0x100) *…continued*

Bit	Symbol	Value	Description	Reset value
26	MAILBOX_RST		Inter CPU communication mailbox reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
27	ADC_RST		ADC reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
31:28	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.8 Peripheral reset control 1

The PRESETCTRL1 register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Remark: It is recommended that changes to the PRESETCTRL registers be accomplished by using the related PRESETCTRLSET and PRESETCTRLCLR registers. It avoids any unintentional setting or clearing of other bits.

Table 46. Peripheral reset control 1 (PRESETCTRL1, offset = 0x104)

Bit	Symbol	Value	Description	Reset value
0	MRT_RST		MRT reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
1	OSTIMER_RST		OS Event Timer reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
2	SCT_RST		SCT0 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
9:3			Reserved. Read value is undefined, only zero should be written.	undefined
10	UTICK_RST		UTICK reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
11	FC0_RST		FC0 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
12	FC1_RST		FC1 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
13	FC2_RST		FC2 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	

Table 46. Peripheral reset control 1 (PRESETCTRL1, offset = 0x104) *…continued*

Bit	Symbol	Value	Description	Reset value
14	FC3_RST		FC3 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
15	FC4_RST		FC4 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
16	FC5_RST		FC5 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
17	FC6_RST		FC6 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
18	FC7_RST		FC7 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
21:19	-		Reserved. Read value is undefined, only zero should be written.	undefined
22	TIMER2_RST		Timer 2 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
24:23	-		Reserved. Read value is undefined, only zero should be written.	undefined
25	USB0_DEV_RST		USB0 device controller reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
26	TIMER0_RST		Timer 0 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
27	TIMER1_RST		Timer 1 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
31:28	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.9 Peripheral reset control 2

The PRESETCTRL2 register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Remark: It is recommended that changes to the PRESETCTRL registers be accomplished by using the related PRESETCTRLSET and PRESETCTRLCLR registers. It avoids any unintentional setting or clearing of other bits.

Table 47. Peripheral reset control 2 (PRESETCTRL2, offset = 0x108)

Bit	Symbol	Value	Description	Reset value
0	-		Reserved. Read value is undefined, only zero should be written.	undefined
1	DMA1_RST		DMA1 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
2	COMP_RST		Analog Comparator reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
3	SDIO_RST		SDIO reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
4	USB1_HOST_RST		USB1 host controller reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
5	USB1_DEV_RST		USB1 device controller reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
6	USB1_RAM_RST		USB1 RAM controller reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
7	USB1_PHY_RST		USB1_PHY reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
8	FREQME_RST		Frequency meter reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
12:9	-		Reserved. Read value is undefined, only zero should be written.	undefined
13	RNG_RST		RNG reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
12:9	-		Reserved. Read value is undefined, only zero should be written.	undefined
15	SYSCTL_RST		SYSCTL Block reset.	0x0
		1	Block is reset.	
		0	Block is not reset.	
16	USB0_HOSTM_RST		USB0 host controller master reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
17	USB0_HOSTS_RST		USB0 host controller slave reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	

Table 47. Peripheral reset control 2 (PRESETCTRL2, offset = 0x108) *…continued*

Bit	Symbol	Value	Description	Reset value
18	HASH_AES_RST		HASH AES reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
19	PQ_RST		Power quad reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
20	PLULUT_RST		PLU LUT reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
21	TIMER3_RST		Timer 3 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
22	TIMER4_RST		Timer 4 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
23	PUF_RST		PUF reset control reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
24	CASPER_RST		Casper reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
25	-		Reserved. Read value is undefined, only zero should be written.	undefined
26	-		Reserved. Read value is undefined, only zero should be written.	undefined
27	ANALOG_CTRL_RST		Analog control reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
28	HS_LSPI_RST		High-Speed SPI reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
29	GPIO_SEC_RST		GPIO secure reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
30	GPIO_SEC_INT_RST		GPIO secure int reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
31	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.10 Peripheral reset control set register0

Writing a 1 to a bit position in PRESETCTRLSET0 sets the corresponding position in PRESETCTRL0. It is a write-only register. For bit assignments, see Table 45.

Table 48. Peripheral reset control register (PRESETCTRLSET0, offset = 0x120)

Bit	Symbol	Description	Reset value
31:0	DATA	Data array value.	0x0

4.5.11 Peripheral reset control set register1

Writing a 1 to a bit position in PRESETCTRLSET1 sets the corresponding position in PRESETCTRL1. It is a write-only register. For bit assignments, see Table 46.

Table 49. Peripheral reset control register (PRESETCTRLSET1, offset = 0x124)

Bit	Symbol	Description	Reset value
31:0	DATA	Data array value.	0x0

4.5.12 Peripheral reset control set register2

Writing a 1 to a bit position in PRESETCTRLSET2 sets the corresponding position in PRESETCTRL2. It is a write-only register. For bit assignments, see Table 47.

Table 50. Peripheral reset control register (PRESETCTRLSET2, offset = 0x128)

Bit	Symbol	Description	Reset value
31:0	DATA	Data array value.	0x0

4.5.13 Peripheral reset control clear register0

Writing a 1 to a bit position in PRESETCTRLCLR0 clears the corresponding position in PRESETCTRL0. This is a write-only register. For bit assignments, see Table 45.

Table 51. Peripheral reset control register (PRESETCTRLCLR0, offset = 0x140)

Bit	Symbol	Description	Reset value
31:0	DATA	Data array value.	0x0

4.5.14 Peripheral reset control clear register1

Writing a 1 to a bit position in PRESETCTRLCLR1 clears the corresponding position in PRESETCTRL1. It is a write-only register. For bit assignments, see Table 46.

Table 52. Peripheral reset control register (PRESETCTRLCLR1, offset = 0x144)

Bit	Symbol	Description	Reset value
31:0	DATA	Data array value.	0x0

4.5.15 Peripheral reset control clear register2

Writing a 1 to a bit position in PRESETCTRLCLR2 clears the corresponding position in PRESETCTRL2. It is a write-only register. For bit assignments, see Table 47.

Table 53. Peripheral reset control register (PRESETCTRLCLR2, offset = 0x148)

Bit	Symbol	Description	Reset value
31:0	DATA	Data array value.	0x0

4.5.16 Software reset register

Write 0x5A00_0001 to generate a software reset.

Table 54. Generate a software reset (SWR_RESET, offset = 0x160)

Bit	Symbol	Value	Description	Reset value
31:0	SWR_RESET		Write 0x5A00_0001 to generate a software_reset.	0x0
		0x5A00_0001	Generate a software reset.	
			All other have values have no effect	

4.5.17 AHB clock control 0

The AHBCLKCTRL0 register enables the clocks to individual system and peripheral blocks. The system clock (bit 0) provides the clock for the AHB, the APB bridges, the CPUs, the SYSCON block, and the PMU. This clock cannot be disabled.

Remark: Use the related AHBCLKCTRLSET and AHBCLKCTRLCLR registers to make changes to the AHBCLKCTRL register to avoid any unintentional setting or clearing of other bits.

See Section 2.1.5 “Memory map overview” for details of SRAM configuration for bits 3, 4, 5, and 6.

Table 55. AHB Clock control 0 (AHBCLKCTRL0, offset = 0x200)

Bit	Symbol	Access	Value	Description	Reset value
0		WO		Reserved. Read value is undefined, only zero should be written.	undefined
1	ROM	RW		Enables the clock for the ROM.	0x0
			1	Enable Clock.	
			0	Disable Clock.	
2	-	WO		Reserved. Read value is undefined, only zero should be written.	undefined
3	SRAM_CTRL1	RW		Enables the clock for the SRAM Controller 1.	0x0
			1	Enable Clock.	
			0	Disable Clock.	
4	SRAM_CTRL2	RW		Enables the clock for the SRAM Controller 2.	0x0
			1	Enable Clock.	
			0	Disable Clock.	
5	SRAM_CTRL3	RW		Enables the clock for the SRAM Controller 3.	0x0
			1	Enable Clock.	
			0	Disable Clock.	
6	SRAM_CTRL4	RW		Enables the clock for the SRAM Controller 4.	0x0
			1	Enable Clock.	
			0	Disable Clock.	
7	FLASH	RW		Enables the clock for the Flash controller.	0x1
			1	Enable Clock.	
			0	Disable Clock.	
8	FMC	RW		Enables the clock for the FMC controller.	0x1
			1	Enable Clock.	
			0	Disable Clock.	
10:9	-	WO		Reserved. Read value is undefined, only zero should be written.	undefined

Table 55. AHB Clock control 0 (AHBCLKCTRL0, offset = 0x200) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
11	MUX	RW		Enables the clock for the Input Mux.	0x0
			1	Enable clock.	
			0	Disable clock.	
12	-	WO		Reserved. Read value is undefined, only zero should be written.	undefined
13	IOCON	RW		Enables the clock for the I/O controller.	0x0
			1	Enable clock.	
			0	Disable clock.	
14	GPIO0	RW		Enables the clock for the GPIO0.	0x0
			1	Enable clock.	
			0	Disable clock.	
15	GPIO1	RW		Enables the clock for the GPIO1.	0x0
			1	Enable clock.	
			0	Disable clock.	
16	GPIO2	RW		Enables the clock for the GPIO2.	0x0
			1	Enable clock.	
			0	Disable clock.	
17	GPIO3	RW		Enables the clock for the GPIO3.	0x0
			1	Enable clock.	
			0	Disable clock.	
18	PINT	RW		Enables the clock for the Pin interrupt (PINT).	0x0
			1	Enable clock.	
			0	Disable clock.	
19	GINT	RW		Enables the clock for the Group interrupt (GINT).	0x0
			1	Enable clock.	
			0	Disable clock.	
20	DMA0	RW		Enables the clock for the DMA0.	0x0
			1	Enable clock.	
			0	Disable clock.	
21	CRCGEN	RW		Enables the clock for the CRCGEN.	0x0
			1	Enable clock.	
			0	Disable clock.	
22	WWDT	RW		Enables the clock for the Watchdog Timer.	0x0
			1	Enable clock.	
			0	Disable clock.	
23	RTC	RW		Enables the clock for the Real Time Clock (RTC).	0x0
			1	Enable clock.	
			0	Disable clock.	
25:24	-	WO		Reserved. Read value is undefined, only zero should be written.	undefined
26	MAILBOX	RW		Enables the clock for the Inter CPU communication Mailbox.	0x0
			1	Enable clock.	
			0	Disable clock.	

Table 55. AHB Clock control 0 (AHBCLKCTRL0, offset = 0x200) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
27	ADC	RW		Enables the clock for the ADC.	0x0
			1	Enable clock.	
			0	Disable clock.	
31:28		WO		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.18 AHB clock control 1

The AHBCLKCTRL1 register enables the clocks to individual peripheral blocks.

Table 56. AHB clock control 1 (AHBCLKCTRL1, offset = 0x204)

Bit	Symbol	Value	Description	Reset value
0	MRT		Enables the clock for the MRT.	0x0
		1	Enable clock.	
		0	Disable clock.	
1	OSTIMER		Enables the clock for the OS Event Timer.	0x0
		1	Enable clock.	
		0	Disable clock.	
2	SCT		Enables the clock for the SCT.	0x0
		1	Enable clock.	
		0	Disable clock.	
9:3	-		Reserved. Read value is undefined, only zero should be written.	undefined
10	UTICK		Enables the clock for the UTICK.	0x0
		1	Enable clock.	
		0	Disable clock.	
11	FC0		Enables the clock for the FC0.	0x0
		1	Enable clock.	
		0	Disable clock.	
12	FC1		Enables the clock for the FC1.	0x0
		1	Enable clock.	
		0	Disable clock.	
13	FC2		Enables the clock for the FC2.	0x0
		1	Enable clock.	
		0	Disable clock.	
14	FC3		Enables the clock for the FC3.	0x0
		1	Enable clock.	
		0	Disable clock.	
15	FC4		Enables the clock for the FC4.	0x0
		1	Enable clock.	
		0	Disable clock.	
16	FC5		Enables the clock for the FC5.	0x0
		1	Enable clock.	
		0	Disable clock.	

Table 56. AHB clock control 1 (AHBCLKCTRL1, offset = 0x204) *…continued*

Bit	Symbol	Value	Description	Reset value
17	FC6		Enables the clock for the FC6.	0x0
		1	Enable clock.	
		0	Disable clock.	
18	FC7		Enables the clock for the FC7.	0x0
		1	Enable clock.	
		0	Disable clock.	
21:19	-		Reserved. Read value is undefined, only zero should be written.	undefined
22	TIMER2		Enables the clock for the Timer 2.	0x0
		1	Enable clock.	
		0	Disable clock.	
24:23	-		Reserved. Read value is undefined, only zero should be written.	undefined
25	USB0_DEV [1]		Enables the clock for the USB0 device controller master and slave interfaces.	0x0
		1	Enable clock.	
		0	Disable clock.	
26	TIMER0		Enables the clock for the Timer 0.	0x0
		1	Enable clock.	
		0	Disable clock.	
27	TIMER1		Enables the clock for the Timer 1.	0x0
		1	Enable clock.	
		0	Disable clock.	
31:28	-		Reserved. Read value is undefined, only zero should be written.	undefined

[1] For VFBGA59 package parts, make sure USB0_DEV bit is cleared to save power.

4.5.19 AHB clock control 2

The AHBCLKCTRL2 register enables the clocks to individual peripheral blocks.

Table 57. AHB clock control 2 (AHBCLKCTRL2, offset = 0x208)

Bit	Symbol	Value	Description	Reset value
0	-		Reserved. Read value is undefined, only zero should be written.	undefined
1	DMA1		Enables the clock for the DMA1.	0x0
		1	Enable clock.	
		0	Disable clock.	
2	COMP		Enables the clock for the Analog Comparator.	0x0
		1	Enable clock.	
		0	Disable clock.	
3	SDIO		Enables the clock for the SDIO.	0x0
		1	Enable clock.	
		0	Disable clock.	
4	USB1_HOST		Enables the clock for the USB1 host controller slave interface.	0x0
		1	Enable clock.	
		0	Disable clock.	

Table 57. AHB clock control 2 (AHBCLKCTRL2, offset = 0x208) *…continued*

Bit	Symbol	Value	Description	Reset value
5	USB1_DEV		Enables the clock for the USB1 device controller slave interface.	0x0
		1	Enable clock.	
		0	Disable clock.	
6	USB1_RAM		Enables the clock for the USB1 RAM controller its slave interface.	0x0
		1	Enable clock.	
		0	Disable clock.	
7	USB1_PHY		Enables the clock for the USB1_PHY APB interface.	0x0
		1	Enable clock.	
		0	Disable clock.	
8	FREQME		Enables the clock for the frequency meter.	0x0
		1	Enable clock.	
		0	Disable clock.	
12:9	-		Reserved. Read value is undefined, only zero should be written.	undefined
13	RNG		Enables the clock for the RNG.	0x0
		1	Enable clock.	
		0	Disable clock.	
14	-		Reserved. Read value is undefined, only zero should be written.	undefined
15	SYSCTL		SYSCTL block clock.	0x0
		1	Enable clock.	
		0	Disable clock.	
16	USB0_HOSTM [1]		Enables the clock for the USB0 host controller master interface.	0x0
		1	Enable clock.	
		0	Disable clock.	
17	USB0_HOSTS [1]		Enables the clock for the USB0 host controller Slave interface.	0x0
		1	Enable clock.	
		0	Disable clock.	
18	HASH_AES		Enables the clock for the HASH_AES.	0x0
		1	Enable clock.	
		0	Disable clock.	
19	PQ		Enables the clock for the power quad.	0x0
		1	Enable clock.	
		0	Disable clock.	
20	PLULUT		Enables the clock for the PLU LUT.	0x0
		1	Enable clock.	
		0	Disable clock.	
21	TIMER3		Enables the clock for the Timer 3.	0x0
		1	Enable clock.	
		0	Disable clock.	
22	TIMER4		Enables the clock for the Timer 4.	0x0
		1	Enable clock.	
		0	Disable clock.	

Table 57. AHB clock control 2 (AHBCLKCTRL2, offset = 0x208) *…continued*

Bit	Symbol	Value	Description	Reset value
23	PUF		Enables the clock for the PUF reset control.	0x0
		1	Enable clock.	
		0	Disable clock.	
24	CASPER		Enables the clock for the Casper.	0x0
		1	Enable clock.	
		0	Disable clock.	
26:25	-		Reserved. Read value is undefined, only zero should be written.	undefined
27	ANALOG_CTRL		Enables the clock for the analog control.	0x0
		1	Enable clock.	
		0	Disable clock.	
28	HS_LSPI		Enables the clock for the High-Speed SPI.	0x0
		1	Enable clock.	
		0	Disable clock.	
29	GPIO_SEC		Enables the clock for the GPIO secure.	0x0
		1	Enable clock.	
		0	Disable clock.	
30	GPIO_SEC_INT		Enables the clock for the GPIO secure interrupt.	0x0
		1	Enable clock.	
		0	Disable clock.	
31	-		Reserved. Read value is undefined, only zero should be written.	undefined

[1] For VFBGA59 package parts, make sure USB0_HOSTM and USB0_HOSTS bits are cleared to save power.

4.5.20 AHB clock control set register 0

Writing a 1 to a bit position in AHBCLKCTRLSET0 sets the corresponding position in AHBCLKCTRL0. It is a write-only register. For bit assignments, see Table 55.

Table 58. Peripheral reset control register (AHBCLKCTRLSET0, offset = 0x220)

Bit	Symbol	Description	Reset value
31:0	DATA	Data array value.	0x0

4.5.21 AHB clock control set register 1

Writing a 1 to a bit position in AHBCLKCTRLSET1 sets the corresponding position in AHBCLKCTRL1. It is a write-only register. For bit assignments, see Table 56.

Table 59. Peripheral reset control register (AHBCLKCTRLSET1, offset = 0x224)

Bit	Symbol	Description	Reset value
31:0	DATA	Data array value.	0x0

4.5.22 AHB clock control set register 2

Writing a 1 to a bit position in AHBCLKCTRLSET2 sets the corresponding position in AHBCLKCTRL2. It is a write-only register. For bit assignments, see Table 57.

Table 60. Peripheral reset control register (AHBCLKCTRLSET2, offset = 0x228)

Bit	Symbol	Description	Reset value
31:0	DATA	Data array value.	0x0

4.5.23 AHB clock control clear register 0

Writing a 1 to a bit position in AHBCLKCTRLCLR0 clears the corresponding position in AHBCLKCTRL0. It is a write-only register. For bit assignments, see Table 55.

Table 61. Peripheral reset control register (AHBCLKCTRLCLR0, offset = 0x240)

Bit	Symbol	Description	Reset value
31:0	DATA	Data array value.	0x0

4.5.24 AHB clock control clear register 1

Writing a 1 to a bit position in AHBCLKCTRLCLR1 clears the corresponding position in AHBCLKCTRL1. It is a write-only register. For bit assignments, see Table 56.

Table 62. Peripheral reset control register (AHBCLKCTRLCLR1, offset = 0x244)

Bit	Symbol	Description	Reset value
31:0	DATA	Data array value.	0x0

4.5.25 AHB clock control clear register 2

Writing a 1 to a bit position in AHBCLKCTRLCLR2 clears the corresponding position in AHBCLKCTRL2. It is a write-only register. For bit assignments, see Table 57.

Table 63. Peripheral reset control register (AHBCLKCTRLCLR2, offset = 0x248)

Bit	Symbol	Description	Reset value
31:0	DATA	Data array value.	0x0

4.5.26 System Tick Timer for CPU0 source select

System Tick Clock for CPU0 comes from the main clock, which is set with register MAINCLKSEL, divided by a rate that is set with register SYSTICKCLKDIV.

Table 64. System Tick Timer for CPU0 source select (SYSTICKCLKSEL0, offset = 0x260)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		System Tick Timer for CPU0 source select.	0x7
		0	System Tick 0 divided clock.	
		1	FRO 1MHz clock.	
		2	Oscillator 32 kHz clock.	
		3	No clock.	
		4	No clock.	
		5	No clock.	
		6	No clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.27 System Tick Timer for CPU1 source select

System Tick Clock for CPU1 comes from the main clock, which is set with register MAINCLKSEL, divided by a rate that is set with register SYSTICKCLKDIV.

Table 65. System Tick Timer for CPU1 source select (SYSTICKCLKSEL1, offset = 0x264)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		System Tick Timer for CPU1 source select.	0x7
		0	System Tick 1 divided clock.	
		1	FRO 1MHz clock.	
		2	Oscillator 32 kHz clock.	
		3	No clock.	
		4	No clock.	
		5	No clock.	
		6	No clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.28 Trace clock source select register

This register selects the clock source for trace clock for CPUs.

Table 66. Trace clock source select (TRACECLKSEL, offset = 0x268)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Trace clock source select.	0x7
		0	Trace divided clock.	
		1	FRO 1 MHz clock.	
		2	Oscillator 32 kHz clock.	
		3	No clock.	
		4	No clock.	
		5	No clock.	
		6	No clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.29 CTimer 0 clock source select

This register selects the clock source for CTimer 0.

Table 67. CTimer 0 clock source select (CTIMERCLKSEL0, offset = 0x26C)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		CTimer 0 clock source select.	0x7
		0	Main clock.	
		1	PLL0 clock.	
		2	No clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	MCLK clock.	
		6	Oscillator 32kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.30 CTimer 1 clock source select register

This register selects the clock source for CTimer 1.

Table 68. CTimer 1 clock source select (CTIMERCLKSEL1, offset = 0x270)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		CTimer 1 clock source select.	0x7
		0	Main clock.	
		1	PLL0 clock.	
		2	No clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	MCLK clock.	
		6	Oscillator 32kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.31 CTimer 2 clock source select register

This register selects the clock source for the CTimer 2.

Table 69. CTimer 2 clock source select (CTIMERCLKSEL2, offset = 0x274)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		CTimer 2 clock source select.	0x7
		0	Main clock.	
		1	PLL0 clock.	
		2	No clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	MCLK clock.	
		6	Oscillator 32kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.32 CTimer 3 clock source select register

This register selects the clock source for CTimer 3.

Table 70. CTimer 3 clock source select (CTIMERCLKSEL3, offset = 0x278)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		CTimer 3 clock source select.	0x7
		0	Main clock.	
		1	PLL0 clock.	
		2	No clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	MCLK clock.	
		6	Oscillator 32kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.33 CTimer 4 clock source select register

This register selects the clock source for CTimer 4.

Table 71. CTimer 4 clock source select (CTIMERCLKSEL4, offset = 0x27C)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		CTimer 4 clock source select.	0x7
		0	Main clock.	
		1	PLL0 clock.	
		2	No clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	MCLK clock.	
		6	Oscillator 32kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.34 Main clock source select register A

This register selects one of the internal oscillator (FRO, or low power oscillator) or an external clock. The oscillator selected is then one of the inputs to the main clock source select register B, see Table 73, which selects the clock source for the main clock. All clocks to the core, memories, and peripherals on the synchronous APB bus are derived from the main clock.

Remark: This selection is internally synchronized; the clock being switched from and the clock being switched to must be running and have occurred in specific states before the selection actually changes.

Table 72. Main clock A source select (MAINCLKSELA, offset=0x280)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Main clock A source select.	0x0
		0	FRO 12 MHz clock.	
		1	CLKIN clock.	
		2	FRO 1MHz clock.	
		3	FRO 96 MHz clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.35 Main clock source select register B

This register selects the clock source for the main clock. All clocks to the core, memories, and peripherals are derived from the main clock.

One input to this register is the main clock source select register A, see Table 72, which selects one of the internal oscillators (FRO, low power oscillator) or an external clock.

Remark: This selection is internally synchronized; the clock being switched from and the clock being switched to must be running and have occurred in specific states before the selection actually changes.

Table 73. Main clock B source select (MAINCLKSELB, offset = 0x284)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Main clock source select.	0x0
		0	Main clock A.	
		1	PLL0 clock.	
		2	PLL1 clock.	
		3	Oscillator 32 kHz clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.36 CLKOUT clock source select register A

This register selects one of the internal oscillators for the clock sources visible on the CLKOUT pin.

Table 74. CLKOUT clock source select (CLKOUTSEL, offset = 0x288)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		CLKOUT clock source select.	0x7
		0	Main clock.	
		1	PLL0 clock.	
		2	CLKIN clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	PLL1 clock.	
		6	Oscillator 32 kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.37 PLL0 clock source select register

This register selects the clock source for the PLL0.

Table 75. PLL0 clock source select (PLL0CLKSEL, offset = 0x290)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		PLL0 clock source select.	0x7
		0	FRO 12 MHz clock.	
		1	CLKIN clock.	
		2	FRO 1 MHz clock.	
		3	Oscillator 32 kHz clock.	
		4	No clock.	
		5	No clock.	
		6	No clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.38 PLL1 clock source select register

This register selects the clock source for PLL1.

Table 76. PLL1 clock source select (PLL1CLKSEL, offset = 0x294)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		PLL1 clock source select.	0x7
		0	FRO 12 MHz clock.	
		1	CLKIN clock.	
		2	FRO 1 MHz clock.	
		3	Oscillator 32 kHz clock.	
		4	No clock.	
		5	No clock.	
		6	No clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.39 ADC clock source select register

This register selects a clock source for the 16-bit ADC that is different from the system clock. To use a clock other than the main clock.

Table 77. ADC clock source select (ADCCLKSEL, offset = 0x2A4)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		ADC clock source select.	0x7
		0	Main clock.	
		1	PLL0 clock.	
		2	FRO 96 MHz clock.	
		3	Reserved.	
		4	No clock.	
		5	No clock.	
		6	No clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.40 USB0 clock source select register

This register selects a clock source for the USB0 device.

Table 78. USB0 clock source select (USB0CLKSEL, offset = 0x2A8)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		USB0 clock source select.	0x7
		0	Main clock.	
		1	PLL0 clock.	
		2	No clock.	
		3	FRO 96 MHz clock.	
		4	No clock.	
		5	PLL1 clock.	
		6	No clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.41 Flexcomm Interface clock source select registers

These registers select the clock source for each Flexcomm Fractional Rate Divider. Each Flexcomm Interface has its own clock source selection and Fractional Rate Divider.

Table 79. Flexcomm Interface 0 clock source select for Fractional Rate Divider (FCCLKSEL0, offset = 0x2B0)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Interface 0 clock source select for Fractional Rate Divider.	0x7
		0	Main clock.	
		1	System PLL divided clock.	
		2	FRO 12 MHz clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	MCLK clock.	
		6	Oscillator 32 kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 80. Flexcomm Interface 1 clock source select for Fractional Rate Divider (FCCLKSEL1, offset = 0x2B4)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Interface 1 clock source select for Fractional Rate Divider.	0x7
		0	Main clock.	
		1	System PLL divided clock.	
		2	FRO 12 MHz clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	MCLK clock.	
		6	Oscillator 32 kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 81. Flexcomm Interface 2 clock source select for Fractional Rate Divider (FCCLKSEL2, offset = 0x2B8)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Interface 2 clock source select for Fractional Rate Divider.	0x7
		0	Main clock.	
		1	System PLL divided clock.	
		2	FRO 12 MHz clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	MCLK clock.	
		6	Oscillator 32 kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 82. Flexcomm Interface 3 clock source select for Fractional Rate Divider (FCCLKSEL3, offset = 0x2BC)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Interface 3 clock source select for Fractional Rate Divider.	0x7
		0	Main clock.	
		1	System PLL divided clock.	
		2	FRO 12 MHz clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	MCLK clock.	
		6	Oscillator 32 kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 83. Flexcomm Interface 4 clock source select for Fractional Rate Divider (FCCLKSEL4, offset = 0x2C0)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Interface 4 clock source select for Fractional Rate Divider.	0x7
		0	Main clock.	
		1	System PLL divided clock.	
		2	FRO 12 MHz clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	MCLK clock.	
		6	Oscillator 32 kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 84. Flexcomm Interface 5 clock source select for Fractional Rate Divider (FCCLKSEL5, offset = 0x2C4)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Interface 5 clock source select for Fractional Rate Divider.	0x7
		0	Main clock.	
		1	System PLL divided clock.	
		2	FRO 12 MHz clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	MCLK clock.	
		6	Oscillator 32 kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 85. Flexcomm Interface 6 clock source select for Fractional Rate Divider (FCCLKSEL6, offset = 0x2C8)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Interface 6 clock source select for Fractional Rate Divider.	0x7
		0	Main clock.	
		1	System PLL divided clock.	
		2	FRO 12 MHz clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	MCLK clock.	
		6	Oscillator 32 kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 86. Flexcomm Interface 7 clock source select for Fractional Rate Divider (FCCLKSEL7, offset = 0x2CC)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Interface 7 clock source select for Fractional Rate Divider.	0x7
		0	Main clock.	
		1	System PLL divided clock.	
		2	FRO 12 MHz clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	MCLK clock.	
		6	Oscillator 32 kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.42 HS SPI clock source select register

This register select the clock source for High-Speed SPI interface.

Table 87. HS SPI clock source select (HSLSPICLKSEL, offset = 0x2D0)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		HS SPI clock source select.	0x7
		0	Main clock.	
		1	System PLL divided clock.	
		2	FRO 12 MHz clock.	
		3	FRO 96 MHz clock.	
		4	FRO 1 MHz clock.	
		5	No clock.	
		6	Oscillator 32 kHz clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.43 I2S MCLK clock source select register

This register selects a clock to provide to the I2S MCLK output function. In a system using I2S and/or digital microphone, this should be related to the clock used by those functions.

Table 88. I2S MCLK clock source select (MCLKCLKSEL, offset = 0x2E0)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		MCLK clock source select.	0x7
		0	FRO 96 MHz clock.	
		1	PLL0 clock.	
		2	Reserved.	
		3	Reserved.	
		4	No clock.	
		5	No clock.	
		6	No clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.44 SCTimer/PWM clock source select register

This register selects a clock to provide to the SC Timer/PWM.

Table 89. SCTimer/PWM clock source select (SCTCLKSEL, offset = 0x2F0)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		SCTimer/PWM clock source select.	0x7
		0	Main clock.	
		1	PLL0 clock.	
		2	CLKIN clock.	
		3	FRO 96 MHz clock.	
		4	No clock.	
		5	MCLK clock.	
		6	No clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.45 SDIO clock source select register

This register selects a clock to provide to the SDIO.

Table 90. SDIO clock source select (SDIOCLKSEL, offset = 0x2F8)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		SDIO clock source select.	0x7
		0	Main clock.	
		1	PLL0 clock.	
		2	No clock.	
		3	FRO 96 MHz clock.	
		4	No clock.	
		5	PLL1 clock.	
		6	No clock.	
		7	No clock.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.46 SYSTICK clock divider register 0

This register configures the SYSTICK divider clock for CPU0.

Table 91. System Tick Timer divider for CPU0 (SYSTICKCLKDIV0, offset = 0x300)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Clock divider value. 0: Divide by 1 ... 255: Divide by 256	0x0
28:8	-		Reserved. Read value is undefined, only zero should be written.	undefined
29	RESET		Resets the divider counter.	0x0
		1	Divider is reset.	
		0	Divider is not reset.	
30	HALT		Halts the divider counter.	0x1
		1	Divider clock is stopped.	
		0	Divider clock is running.	
31	REQFLAG		Divider status flag.	0x0
		1	Clock frequency is not stable.	
		0	Divider clock is stable.	

4.5.47 SYSTICK clock divider register 1

This register configures the SYSTICK divider clock for CPU1.

Table 92. System Tick Timer divider for CPU1 (SYSTICKCLKDIV1, offset = 0x304)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Clock divider value. 0: Divide by 1 ... 255: Divide by 256	0x0
28:8	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 92. System Tick Timer divider for CPU1 (SYSTICKCLKDIV1, offset = 0x304) *…continued*

Bit	Symbol	Value	Description	Reset value
29	RESET		Resets the divider counter.	0x0
		1	Divider is reset.	
		0	Divider is not reset.	
30	HALT		Halts the divider counter.	0x1
		1	Divider clock is stopped.	
		0	Divider clock is running.	
31	REQFLAG		Divider status flag.	0x0
		1	Clock frequency is not stable.	
		0	Divider clock is stable.	

4.5.48 Trace clock divider register

This register configures the trace clock, which is used in conjunction with SWO during debug.

Table 93. TRACE clock divider (TRACECLKDIV, offset = 0x308)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Clock divider value. 0: Divide by 1 ... 255: Divide by 256	0x0
28:8	-		Reserved. Read value is undefined, only zero should be written.	undefined
29	RESET		Resets the divider counter.	0x0
		1	Divider is reset.	
		0	Divider is not reset.	
30	HALT		Halts the divider counter.	0x1
		1	Divider clock is stopped.	
		0	Divider clock is running.	
31	REQFLAG		Divider status flag.	0x0
		1	Clock frequency is not stable.	
		0	Divider clock is stable.	

4.5.49 Fractional rate divider for each Flexcomm Interface frequency

At each Flexcomm Interface, the frequency can be adjusted by a fractional divider.This is primarily to create a base baud rate clock for USART functions, but can be used for other purposes. Each Flexcomm interface has a dedicated register that sets the MULT and DIV values for the fractional rate generator.

The FRG maximum allowed output frequency is **100 MHz**.

Flexcomm Interface function clock = (clock selected via FCCLKSEL) / (1+ MULT /DIV)

The clock used by the fractional rate generator is selected via the FCCLKSEL register (see Section 4.5.41 “Flexcomm Interface clock source select registers”).

Remark: To use the fractional baud rate generator, 0xFF must be wirtten to the DIV value to yield a denominator vale of 256. All other values are not supported. See Section 34.3.1 “Configure the Flexcomm Interface clock and USART baud rate” and Section 34.7.2 “Clocking and baud rates”.

Table 94. Fractional rate divider for Flexcomm Interface 0 (FLEXFRG0CTRL, offset = 0x320)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Denominator of the Fractional Rate Divider.	0xFF
15:8	MULT		Numerator of the Fractional Rate Divider.	0x0
31:16	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 95. Fractional rate divider for Flexcomm Interface 1 (FLEXFRG1CTRL, offset = 0x324)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Denominator of the fractional rate divider.	0xFF
15:8	MULT		Numerator of the fractional rate divider.	0x0
31:16	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 96. Fractional rate divider for Flexcomm Interface 2 (FLEXFRG2CTRL, offset = 0x328)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Denominator of the fractional rate divider.	0xFF
15:8	MULT		Numerator of the fractional rate divider.	0x0
31:16	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 97. Fractional rate divider for Flexcomm Interface 3 (FLEXFRG3CTRL, offset = 0x32C)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Denominator of the fractional rate divider.	0xFF
15:8	MULT		Numerator of the fractional rate divider.	0x0
31:16	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 98. Fractional rate divider for Flexcomm Interface 4 (FLEXFRG4CTRL, offset = 0x330)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Denominator of the fractional rate divider.	0xFF
15:8	MULT		Numerator of the fractional rate divider.	0x0
31:16	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 99. Fractional rate divider for Flexcomm Interface 5 (FLEXFRG5CTRL, offset = 0x334)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Denominator of the fractional rate divider.	0xFF
15:8	MULT		Numerator of the fractional rate divider.	0x0
31:16	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 100. Fractional rate divider for Flexcomm Interface 6 (FLEXFRG6CTRL, offset = 0x338)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Denominator of the fractional rate divider.	0xFF
15:8	MULT		Numerator of the fractional rate divider.	0x0
31:16	-		Reserved. Read value is undefined, only zero should be written.	undefined

Table 101. Fractional rate divider for Flexcomm Interface 7 (FLEXFRG7CTRL, offset = 0x33C)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Denominator of the fractional rate divider.	0xFF
15:8	MULT		Numerator of the fractional rate divider.	0x0
31:16	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.50 AHB clock divider register

This register controls how the main clock is divided to provide the system clock to the AHB bus, CPUs, and memories.

Table 102. System clock divider (AHBCLKDIV, offset = 0x380)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Clock divider value. 0: Divide by 1 ... 255: Divide by 256	0x0
28:8	-		Reserved. Read value is undefined, only zero should be written.	undefined
29	RESET		Resets the divider counter.	0x0
		1	Divider is reset.	
		0	Divider is not reset.	
30	HALT		Halts the divider counter.	0x0
		1	Divider clock is stopped.	
		0	Divider clock is running.	
31	REQFLAG		Divider status flag.	0x0
		1	Clock frequency is not stable.	
		0	Divider clock is stable.	

4.5.51 CLKOUT clock divider register

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

Table 103. CLKOUT clock divider (CLKOUTDIV, offset = 0x384)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Clock divider value. 0: Divide by 1 ... 255: Divide by 256	0x0
28:8	-		Reserved. Read value is undefined, only zero should be written.	undefined
29	RESET		Resets the divider counter.	0x0
		1	Divider is reset.	
		0	Divider is not reset.	
30	HALT		Halts the divider counter.	0x1
		1	Divider clock is stopped.	
		0	Divider clock is running.	
31	REQFLAG		Divider status flag.	0x0
		1	Clock frequency is not stable.	
		0	Divider clock is stable.	

4.5.52 FRO_HF clock divider

This register determines the divider value from the clock signal FRO_HF (the output 96 MHz of the FRO_192) on Flexcomm Interface clocks.

Table 104. FRO_HF clock divider (FROHFDIV, offset = 0x388)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Clock divider value. 0: Divide by 1 ... 255: Divide by 256	0x0
28:8	-		Reserved. Read value is undefined, only zero should be written.	undefined
29	RESET		Resets the divider counter.	0x0
		1	Divider is reset.	
		0	Divider is not reset.	
30	HALT		Halts the divider counter.	0x1
		1	Divider clock is stopped.	
		0	Divider clock is running.	
31	REQFLAG		Divider status flag.	0x0
		1	Clock frequency is not stable.	
		0	Divider clock is stable.	

4.5.53 WWDT clock divider

This register determines the divider value from the clock signal FRO_1 MHz on WDT.

Table 105. WDT clock divider (WDTCLKDIV, offset = 0x38C)

Bit	Symbol	Value	Description	Reset value
5:0	DIV		Clock divider value. 0: Divide by 1. … 63: Divide by 64.	0x0
28:6	-		Reserved. Read value is undefined, only zero should be written.	undefined
29	RESET		Resets the divider counter.	0x0
		1	Divider is reset.	
		0	Divider is not reset.	
30	HALT		Halts the divider counter.	0x1
		1	Divider clock is stopped.	
		0	Divider clock is running.	
31	REQFLAG		Divider status flag.	0x0
		1	Clock frequency is not stable.	
		0	Divider clock is stable.	

4.5.54 ADC clock source divider register

This register divides the clock to the ADC.

Table 106. ADC clock divider (ADCCLKDIV, offset = 0x394)

Bit	Symbol	Value	Description	Reset value
2:0	DIV		Clock divider value. 0: Divide by 1. … 7: Divide by 8	0x0
28:3	-		Reserved. Read value is undefined, only zero should be written.	undefined
29	RESET		Resets the divider counter.	0x0
		1	Divider is reset.	
		0	Divider is not reset.	
30	HALT		Halts the divider counter.	0x1
		1	Divider clock is stopped.	
		0	Divider clock is running.	
31	REQFLAG		Divider status flag.	0x0
		1	Clock frequency is not stable.	
		0	Divider clock is stable.	

4.5.55 USB0 full-speed clock divider register

This register determines the divider value for the USB full-speed function clock.

Table 107. USB0 clock divider (USB0CLKDIV, offset = 0x398)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Clock divider value. 0: Divide by 1. … 255: Divide by 256	0x0
28:8	-		Reserved. Read value is undefined, only zero should be written.	undefined
29	RESET		Resets the divider counter.	0x0
		1	Divider is reset.	
		0	Divider is not reset.	
30	HALT		Halts the divider counter.	0x1
		1	Divider clock is stopped.	
		0	Divider clock is running.	
31	REQFLAG		Divider status flag.	0x0
		1	Clock frequency is not stable.	
		0	Divider clock is stable.	

4.5.56 I²S MCLK clock divider register

This register determines the divider value for the I²S MCLK output, if used by the application.

Table 108. I²S MCLK clock divider (MCLKDIV, offset = 0x3AC)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Clock divider value. 0: Divide by 1. … 255: Divide by 256	0x0
28:8	-		Reserved. Read value is undefined, only zero should be written.	undefined
29	RESET		Resets the divider counter.	0x0
		1	Divider is reset.	
		0	Divider is not reset.	
30	HALT		Halts the divider counter.	0x1
		1	Divider clock is stopped.	
		0	Divider clock is running.	
31	REQFLAG		Divider status flag.	0x0
		1	Clock frequency is not stable.	
		0	Divider clock is stable.	

4.5.57 SCTimer/PWM clock divider

This register determines the divider value for the SCTimer/PWM output, if used by the application.

Table 109. SCTimer/PWM clock divider (SCTCLKDIV, offset = 0x3B4)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Clock divider value. 0: Divide by 1. … 255: Divide by 256	0x0
28:8	-		Reserved. Read value is undefined, only zero should be written.	undefined
29	RESET		Resets the divider counter.	0x0
		1	Divider is reset.	
		0	Divider is not reset.	
30	HALT		Halts the divider counter.	0x1
		1	Divider clock is stopped.	
		0	Divider clock is running.	
31	REQFLAG		Divider status flag.	0x0
		1	Clock frequency is not stable.	
		0	Divider clock is stable.	

4.5.58 SDIO clock divider

This register determines the divider value for the SDIO output, if used by the application.

Table 110. SDIO clock divider (SDIOCLKDIV, offset = 0x3BC)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Clock divider value. 0: Divide by 1. … 255: Divide by 256	0x0
28:8	-		Reserved. Read value is undefined, only zero should be written.	undefined
29	RESET		Resets the divider counter.	0x0
		1	Divider is reset.	
		0	Divider is not reset.	
30	HALT		Halts the divider counter.	0x1
		1	Divider clock is stopped.	
		0	Divider clock is running.	
31	REQFLAG		Divider status flag.	0x0
		1	Clock frequency is not stable.	
		0	Divider clock is stable.	

4.5.59 PLL0 clock divider

This register determines the divider value for the PLL0 output, if used by the application.

Table 111. PLL0 clock divider (PLL0CLKDIV, offset = 0x3C4)

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0x0
28:8	-		Reserved. Read value is undefined, only zero should be written.	undefined
29	RESET		Resets the divider counter.	0x0
		1	Divider is reset.	
		0	Divider is not reset.	
30	HALT		Halts the divider counter.	0x1
		1	Divider clock is stopped.	
		0	Divider clock is running.	
31	REQFLAG		Divider status flag.	0x0
		1	Clock frequency is not stable.	
		0	Divider clock is stable.	

4.5.60 Control clock configuration registers access

This register is used to prevent access to clock select and divider configuration.

Table 112. Control clock configuration registers access (xxxDIV, xxxSEL) (CLOCKGENUPDATELOCKOUT, offset = 0x3FC)

Bit	Symbol	Value	Description	Reset value
31:0	CLOCKGENUPDATELOCKOUT		Control clock configuration registers access (for example, xxxDIV, xxxSEL).	0x0
		1	Update all clock configuration.	
		0	All hardware clock configuration are freeze.	

4.5.61 FMC configuration register

This register controls FMC configuration. Depending on the system clock frequency, access to the flash memory can be configured with various access times by writing to the FMCCR register.

It is recommended to use the power API to configure device operation in order to achieve lower power operation. However, flash timing can also be set up by user software as shown in Table 113. For device revision 1B, the power library in the SDK sets the DCDC output based on the selected frequency. For frequencies below 100 MHz, DCDC output is set at 1.075 V, for frequencies between 100 MHz to 130 MHz, DCDC output is set at 1.15 V, and for frequencies above 150 MHz, DCDC output is set at 1.2 V.

Enabling buffering, acceleration, and prefetch will substantially improve performance. Buffering saves power by allowing previously accessed information to be reused without a flash read. Acceleration saves power by reducing CPU stalls. Prefetch typically has a small power cost due to some flash reads being performed that ultimately are not needed.

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

Table 113. FMC configuration register (FMCCR, offset = 0x400) bit description

Bit	Symbol	Value	Description	Reset value
1:0	FETCHCFG		Instruction fetch configuration. This field determines how flash accelerator buffers are used for instruction fetches.	0x2
		0x0	Instruction fetches from flash are not buffered. Every fetch request from the CPU results in a read of the flash memory. This setting may use significantly more power than when buffering is enabled.	
		0x1	One buffer is used for all instruction fetches.	
		0x2	All buffers may be used for instruction fetches.	
		0x3	Reserved setting, do not use.	
3:2	DATACFG		Data read configuration. This field determines how flash accelerator buffers are used for data accesses.	0x2
		0x0	Data accesses from flash are not buffered. Every data access from the CPU results in a read of the flash memory.	
		0x1	One buffer is used for all data accesses.	
		0x2	All buffers may be used for data accesses.	
		0x3	Reserved setting, do not use.	
4	ACCEL		Acceleration enable.	1
		0	Flash acceleration is disabled. Every flash read (including those fulfilled from a buffer) takes FLASHTIM + 1 system clocks. This allows more determinism at a cost of performance.	
		1	Flash acceleration is enabled. Performance is enhanced, dependent on other FMCCR settings.	
5	PREFEN		Prefetch enable. [1][2]	0
		0	No instruction prefetch is performed.	
		1	If the FETCHCFG field is not 0, the next flash line following the current execution address is automatically prefetched if it is not already buffered.	
6	PREFOVR		Prefetch override. This bit only applies when PREFEN = 1 and a buffered instruction is completing for which the next flash line is not already buffered or being prefetched.	0x0
		0	Any previously initiated prefetch will be completed.	
		1	Any previously initiated prefetch will be aborted, and the next flash line following the current execution address will be prefetched if not already buffered.	
15:7	-	-	Reserved.	-
16:12	FLASHTIM		Flash memory access time. The number of system clocks used for flash accesses is equal to FLASHTIM +1.	0x0
		0x0	1 system clock flash access time (for system clock rates up to 11 MHz).	
		0x1	2 system clocks flash access time (for system clock rates up to 22 MHz).	
		0x2	3 system clocks flash access time (for system clock rates up to 33 MHz).	
		0x3	4 system clocks flash access time (for system clock rates up to 44 MHz).	
		0x4	5 system clocks flash access time (for system clock rates up to 55 MHz).	
		0x5	6 system clocks flash access time (for system clock rates up to 66 MHz).	
		0x6	7 system clocks flash access time (for system clock rates up to 77 MHz).	
		0x7	8 system clocks flash access time (for system clock rates up to 88 MHz).	
		0x8	9 system clocks flash access time (for system clock rates up to 100 MHz).	
		0x9	10 system clocks flash access time (for system clock rates up to 115 MHz).	

Table 113. FMC configuration register (FMCCR, offset = 0x400) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
		0xA	11 system clocks flash access time (for system clock rates up to 130 MHz).	
		0xB	12 system clocks flash access time (for system clock rates up to 150 MHz).	
31:17	-	-	Reserved.	-

[1] The prefetch bit is not functional on device revision 0A. The prefetch bit is functional on device revision 1B.

[2] The prefetch bit must be disabled before executing any flash programming/erasing and flash controller commands.

4.5.62 USB0 need clock control register

This register controls the polarity of the USB0 need clock signals for triggering the USB wake-up interrupt. For details of how to use the register for waking up the part from deep-sleep mode, see Section 41.8.6 “USB0 wake-up”.

Table 114. USB0 need clock control (USB0NEEDCLKCTRL, offset = 0x40C)

Bit	Symbol	Value	Description	Reset value
0	AP_FS_DEV_NEEDCLK		USB0 device need clock signal control.	0x0
		0	DEV_NEEDCLK is Under hardware control.	
		1	DEV_NEEDCLK forced high.	
1	POL_FS_DEV_NEEDCLK		USB0 device need clock polarity for triggering the USB0_NEEDCLK wake-up interrupt.	0x0
		0	Falling edge of DEV_NEEDCLK triggers wake-up.	
		1	Rising edge of DEV_NEEDCLK triggers wake-up.	
2	AP_FS_HOST_NEEDCLK		USB0 Host need clock signal control.	0x0
		0	HOST_NEEDCLK is under hardware control.	
		1	HOST_NEEDCLK is forced high.	
3	POL_FS_HOST_NEEDCLK		USB0 Host need clock polarity for triggering the USB0_NEEDCLK wake-up interrupt.	0x0
		0	Falling edge of device HOST_NEEDCLK triggers wake-up.	
		1	Rising edge of device HOST_NEEDCLK triggers wake-up.	
31:4	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.63 USB0 need clock status register

This register is read-only and returns the status of the device and host need clock signals. For details of how to use this register signal for waking up the part from deep-sleep mode, see Section 41.8.6 “USB0 wake-up”.

Table 115. USB0 need clock status (USB0NEEDCLKSTAT, offset = 0x410)

Bit	Symbol	Value	Description	Reset value
0	DEV_NEEDCLK		USB0 device need clock signal status.	0x0
		1	DEV_NEEDCLK signal is high.	
		0	DEV_NEEDCLK signal is low.	

Table 115. USB0 need clock status (USB0NEEDCLKSTAT, offset = 0x410) *...continued*

Bit	Symbol	Value	Description	Reset value
1	HOST_NEEDCLK		USB0 host need clock signal status.	0x0
		1	HOST_NEEDCLK signal is high.	
		0	HOST_NEEDCLK signal is low.	
31:2	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.64 FMC flush control register

This register is to flush the FMC cache from software. Since the FMC holds data after PRINCE decryption, cache should be flushed when PRINCE IV or keys are updated.

Table 116. FMC flush control (FMCFLUSH, offset = 0x41C)

Bit	Symbol	Value	Description	Reset value
0	FLUSH		Controls flushing the contents of the FMC buffers.	0x0
		0	No action.	
		1	Flush the contents of the FMC buffers, then self-clear to 0.	
31:1	-		Reserved. Only zero should be written.	undefined

4.5.65 MCLKIO control

This register selects the direction of the pin associated with MCLK when MCLK is the elected function on that pin.

Table 117. MCLK control (MCLKIO, offset = 0x420)

Bit	Symbol	Value	Description	Reset value
0	MCLKIO		MLK control.	0x0
		0	Input mode.	
		1	Output mode.	
31:1	-		Reserved.	undefined

4.5.66 USB1 need clock control register

This register controls the polarity of the USB1 need clock signals for triggering the USB USB1_NEEDCLK wake-up interrupt. For details of how to use this register for waking up the part from deep-sleep mode, see Section 44.7.6 “USB1 wake-up”.

Table 118. USB1 need clock control (USB1NEEDCLKCTRL, offset = 0x424)

Bit	Symbol	Value	Description	Reset value
0	AP_HS_DEV_NEEDCLK		USB1 device need clock signal control:.	0x0
		0	HOST_NEEDCLK is under hardware control.	
		1	HOST_NEEDCLK is forced high.	
1	POL_HS_DEV_NEEDCLK		USB1 device need clock polarity for triggering the USB1_NEEDCLK wake-up interrupt:.	0x0
		0	Falling edge of DEV_NEEDCLK triggers wake-up.	
		1	Rising edge of DEV_NEEDCLK triggers wake-up.	

Table 118. USB1 need clock control (USB1NEEDCLKCTRL, offset = 0x424) *…continued*

Bit	Symbol	Value	Description	Reset value
2	AP_HS_HOST_NEEDCLK		USB1 Host need clock signal control:.	0x0
		0	HOST_NEEDCLK is under hardware control.	
		1	HOST_NEEDCLK is forced high.	
3	POL_HS_HOST_NEEDCLK		USB1 host need clock polarity for triggering the USB1_NEEDCLK wake-up interrupt.	0x0
		0	Falling edge of HOST_NEEDCLK triggers wake-up.	
		1	Rising edge of HOST_NEEDCLK triggers wake-up.	
4	HS_DEV_WAKEUP_N		Software override of device controller PHY wake up logic.	0x1
		0	Forces USB1_PHY to wake-up.	
		1	Normal USB1_PHY behavior.	
31:5	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.67 USB1 need clock status register

This register is read-only and returns the status of the device and host need clock signals. For details of how to use this register for waking up the part from deep-sleep mode, see Section 44.7.6 “USB1 wake-up”.

Table 119. USB1 need clock status (USB1NEEDCLKSTAT, offset = 0x428)

Bit	Symbol	Value	Description	Reset value
0	DEV_NEEDCLK		USB1 device need clock signal status.	0x0
		1	DEV_NEEDCLK is high.	
		0	DEV_NEEDCLK is low.	
1	HOST_NEEDCLK		USB1 host need clock signal status.	0x0
		1	HOST_NEEDCLK is high.	
		0	HOST_NEEDCLK is low.	
31:2	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.68 SDIO CCLKIN phase and delay control

This register delays the cclk_in_sample and cclk_in_drc wrt ccclk_in. Both phase and delay shifts are sequential, so if activated together it is cumulative.

Table 120. SDIO CCLKIN phase and delay control (SDIOCLKCTRL, offset = 0x460)

Bit	Symbol	Value	Description	Reset value
1:0	CCLK_DRV_PHASE		Programmable delay value by which cclk_in_drv is phase-shifted with regard to cclk_in.	0x0
		0	0 degree shift.	
		1	90 degree shift.	
		2	180 degree shift.	
		3	270 degree shift.	
3:2	CCLK_SAMPLE_PHASE		Programmable delay value by which cclk_in_sample is delayed with regard to cclk_in.	0x0
		0	0 degree shift.	
		1	90 degree shift.	
		2	180 degree shift.	
		3	270 degree shift.	
6:4	-		Reserved. Read value is undefined, only zero should be written.	undefined
7	PHASE_ACTIVE		Enables the delays CCLK_DRV_PHASE and CCLK_SAMPLE_PHASE.	0x0
		0	Bypassed.	
		1	Activates phase shift logic. When active, the clock divider is active and phase delays are enabled.	
15:8	-		Reserved. Read value is undefined, only zero should be written.	undefined
20:16	CCLK_DRV_DELAY		Programmable delay value by which cclk_in_drv is delayed with regard to cclk_in.	0x0
22:21	-		Reserved. Read value is undefined, only zero should be written.	undefined
23	CCLK_DRV_DELAY_ACTIVE		Enables drive delay, as controlled by the CCLK_DRV_DELAY field.	0x0
		1	Enable drive delay.	
		0	Disable drive delay.	
28:24	CCLK_SAMPLE_DELAY		Programmable delay value by which cclk_in_sample is delayed with regard to cclk_in.	0x0
30:29	-		Reserved. Read value is undefined, only zero should be written.	undefined
31	CCLK_SAMPLE_DELAY_ACTIVE		Enables sample delay, as controlled by the CCLK_SAMPLE_DELAY field.	0x0
		1	Enables sample delay.	
		0	Disables sample delay.	

4.5.69 PLL registers

The PLL registers provide a wide range of frequencies and can potentially be used for many on-chip functions. The PLL registers can be used with or without a spread spectrum clock generator. See Section 4.6.6 “PLL0 and PLL1 functional description” for additional details of PLL operation.

4.5.69.1 PLL0

4.5.69.1.1 PLL0 control register

The PLL0CTRL register provides most of the control over basic selections of PLL0 modes and operating details.

Table 121. PLL0 control (PLL0CTRL, offset = 0x580)

Bit	Symbol	Access	Value	Description	Reset value
3:0	SELR	RW		Bandwidth select R value.	0x0
9:4	SELI	RW		Bandwidth select I value.	0x0
14:10	SELP	RW		Bandwidth select P value.	0x0
15	BYPASSPLL	RW		Bypass PLL input clock is sent directly to the PLL output.	0x0
			1	Bypass PLL input clock is sent directly to the PLL output.	
			0	Use PLL.	
16	BYPASSPOSTDIV2	RW		Bypass of the divide-by-2 divider in the post-divider.	0x0
			1	Bypass of the divide-by-2 divider in the post-divider.	
			0	Use the divide-by-2 divider in the post-divider.	
17	LIMUPOFF	RW		limup_off = 1 in spread spectrum and fractional PLL applications.	0x0
18	BWDIRECT	RW		Control of the bandwidth of the PLL.	0x0
			1	Modify the bandwidth of the PLL directly.	
			0	The bandwidth is changed synchronously with the feedback-divider.	
19	BYPASSPREDIV	RW		Bypass of the pre-divider.	0x0
			1	Bypass of the pre-divider.	
			0	Use the pre-divider.	
20	BYPASSPOSTDIV	RW		Bypass of the post-divider.	0x0
			1	Bypass of the post-divider.	
			0	Use the post-divider.	
21	CLKEN	RW		Enable the output clock.	0x0
			1	Enable the output clock.	
			0	Disable the output clock.	
22	FRMEN	RW		Free running mode.	0x0
			1	Free running mode is enable.	
			0	Free running mode is disable.	
23	FRMCLKSTABLE	RW		Free running mode clock stable. **Note:** frm_clockstable can be =1 only after the PLL output frequency is stable.	0x0

Table 121. PLL0 control (PLL0CTRL, offset = 0x580) *...continued*

Bit	Symbol	Access	Value	Description	Reset value
24	SKEWEN	RW		Skew mode.	0x0
			1	Skew mode is enable.	
			0	Skew mode is disable.	
31:25	-	WO		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.69.1.2 PLL0 status register

The read-only PLL0STAT register provides the PLL lock status and other status details.

Remark: The lock status does not reliably indicate the PLL status for the following two configurations: spread-spectrum mode or fractional enabled or low input clock frequencies such as 32 kHz. In these cases, refer to the PLL lock times listed in the specific device data sheet to obtain appropriate wait times for the PLL to lock.

Table 122. PLL0 status (PLL0STAT, offset = 0x584)

Bit	Symbol	Access	Description	Reset value
0	LOCK	RO	Lock detector output (active high) Warning: The lock signal is only reliable between fref[2]:100 kHz to 20 MHz.	0x0
1	PREDIVACK	RO	Pre-divider ratio change acknowledge.	0x0
2	FEEDDIVACK	RO	Feedback divider ratio change acknowledge.	0x0
3	POSTDIVACK	RO	Post-divider ratio change acknowledge.	0x0
4	FRMDET	RO	Free running detector output (active high).	0x0
31:5	-	WO	Reserved. Read value is undefined, only zero should be written.	undefined

Remark: Please see PLL Lock Bit errata regarding details on ensuring the PLL is stable and locked.

4.5.69.1.3 PLL0 N-divider register

The PLL0NDEC controls operation of the PLL pre-divider.

Table 123. PLL0 N divider (PLL0NDEC, offset = 0x588)

Bit	Symbol	Access	Description	Reset value
7:0	NDIV	RW	Pre-divider, divider ratio (N-divider).	0x0
8	NREQ	RW	Pre-divider ratio change request.	0x0
31:9	-	WO	Reserved. Read value is undefined, only zero should be written.	undefined

4.5.69.1.4 PLL0 P-divider register

The PLL0PDEC controls operation of the PLL post-divider.

Table 124. PLL0 P divider (PLL0PDEC, offset = 0x58C)

Bit	Symbol	Access	Description	Reset value
4:0	PDIV	RW	Post-divider, divider ratio (P-divider).	0x0
5	PREQ	RW	Feedback ratio change request.	0x0
31:6	-	WO	Reserved. Read value is undefined, only zero should be written.	undefined

4.5.69.1.5 Spread spectrum control with the System PLL

The spread spectrum functionality can be used to modulate the PLL output frequency. This can decrease electromagnetic interference (EMI) in an application. The Spread Spectrum Clock Generator can be used in several ways:

- It can encode M-divider values between 1 and 255 to produce the MDEC value used directly by the PLL, saving the need for executing encoding algorithm code, or hard-coding predetermined values into an application.
- It can provide a fractional rate feature to the PLL.
- It can be set up to automatically alter the PLL CCO frequency on an ongoing basis to decrease electromagnetic interference (EMI).

If the spread spectrum mode is enabled, choose N to ensure 3 MHz < Fin/N < 5 MHz. Spread spectrum mode cannot be used when Fin = 32 kHz.

When the modulation (MR) is set to zero, the PLL becomes a fractional PLL. Please refer to section Section 4.6.6.3.3 “Spread spectrum mode”.

PLL0 spread spectrum control register 0

Table 125. PLL0 spread spectrum wrapper control register 0 (PLL0SSCG0, offset = 0x590)

Bit	Symbol	Access	Value	Description	Reset value
31:0	MD_LBS	RW		Input word of the wrapper bits 31 to 0.	0x0

PLL0 spread spectrum control register 1

Table 126. PLL0 spread spectrum wrapper control register 1 (PLL0SSCG1, offset = 0x594)

Bit	Symbol	Access	Description	Reset value
0	MD_MBS	RW	Input word of the wrapper bit 32.	0x0
1	MD_REQ	RW	MD change request.	0x0
4:2	MF	RW	Programmable modulation frequency fm = Fref/Nss mf[2:0] = 000 => Nss=512 (fm 3.	0x0
7:5	MR	RW	Programmable frequency modulation depth: Dfmodpk-pk = Fref kss/Fcco= kss/(2md[32:25]dec)mr[2:0] = 000 => kss = 0 (no spread spectrum) mr[2:0] = 001 => kss 1 mr[2:0] = 010 => kss 1	0x0
9:8	MC	RW	Modulation waveform control Compensation for low pass filtering of the PLL to get a triangular modulation at the output of the PLL, giving a flat frequency spectrum.	0x0
25:10	MDIV_EXT	RW	To select an external mdiv value.	0x0
26	MREQ	RW	To select an external mreq value.	0x0
27	DITHER	RW	dithering between two modulation frequencies in a random way or in a pseudo random way (white noise), in order to decrease the probability that the modulated waveform will occur with the same phase on a particular point on the screen.	0x0
28	SEL_EXT	RW	To select mdiv_ext and mreq_ext sel_ext = 0: mdiv md[32:0], mreq = 1 sel_ext = 1 : mdiv = mdiv_ext, mreq = mreq_ext.	0x0
31:29	-	WO	Reserved. Read value is undefined, only zero should be written.	undefined

Note: MD_MBS+MD_LBS fields represent md[32:0].

Table 127. Modulator input (spread spectrum enabled): md => F_{clkcco}= (md[32:25]dec + md[24:0]dec*2^{-25})*F_{ref}

md[32:0] (min. frequency)	100000000_{hex} => $md[32:25]_{dec}$=16, $md[24:0]_{dec}$=0, F_{clkcco}=2 *16*F_{ref}
md[32:0]	380000001_{hex} => $md[32:25]_{dec}$=56, $md[24:0]_{dec}$=1, F_{clkcco}=2 * (56+2^{-25})*F_{ref}
md[32:0]	380000002_{hex} => $md[32:25]_{dec}$=56, $md[24:0]_{dec}$=2, F_{clkcco}=2 * (56+2^{-24})*F_{ref}
md[32:0]	380000003_{hex} => $md[32:25]_{dec}$=56, $md[24:0]_{dec}$=3, F_{clkcco}=2 * (56+2^{-24}+2^{-25})*F_{ref}
md[32:0]	etcetera
md[32:0] (max. frequency)	$F30000000_{hex}$ => $md[32:25]_{dec}$=243, $md[24:0]_{dec}$=0, F_{clkcco}=2 * (243)*F_{ref}

Table 128. Modulator input (spread spectrum disabled, only fractional => mc=0, mr=0): md => F_{clkcco} = (md[32:25]dec + md[24:0]dec*2^{-25}) * F_{ref}

md[32:0]	md[32:0] = 2^{25}*Maverage => Maverage=($F_{cco,PLL}$ / $F_{ref,PLL}$)
md[32:0] max. value (overflow ΣΔ modulator)	1 1111 0110 0000 0000 0000 0000 0000 0000_{bin} = 1F600000hex => Maverage,max = 251(see Table 129 “Examples of M and stepsize if pd = ‘1’”).
md[32:0] min. value (overflow ΣΔ modulator)	0 0000 1000 0000 0000 0000 0000 0000 0000_{bin} = 08000000hex => Maverage,min = 4 (see Table 129 “Examples of M and stepsize if pd = ‘1’”).

Table 129. Examples of M and stepsize if pd = ‘1’

md[32:0]	**M[1] (pd =”1”)**	**stepsize[2] (pd =”1”)**
0 1000 0000 0000 0000 0000 0000 0000 0000_{bin} = 080000000_{hex} = 64 * 2^{25}	64	1/M»1.56%
0 1000 0010 0000 0000 0000 0000 0000 0000_{bin} = 082000000_{hex} = 65 * 2^{25}	65	1/M»1.54%
0 1000 0010 0000 0000 0000 0000 0000 0001_{bin} = 082000000_{hex} = 65 * 2^{25}+1	65	1/M»1.54%
0 1000 0011 0000 0000 0000 0000 0000 0000_{bin} = 083000000_{hex} = 65 * 2^{25}+2^{24}	65	1/M»1.54%
0 1011 1000 0000 0000 0000 0000 0000 0000_{bin} = $0B8000000_{hex}$ = 92 * 2^{25}	92	1/M»1.09%
0 0011 0110 0000 0000 0000 0000 0000 0000_{bin} = 036000000_{hex} = 27 * 2^{25}	27	1/M»3.70%

[1] [1] M = $md[32:25]_{dec}$

[2] stepsize=1 / M

4.5.69.2 PLL1

4.5.69.2.1 PLL1 control register

The PLL1CTRL register provides most of the control over basic selections of PLL1 modes and operating details.

Table 130. PLL1 control (PLL1CTRL, offset = 0x560)

Bit	Symbol	Access	Value	Description	Reset value
3:0	SELR	RW		Bandwidth select R value.	0x0
9:4	SELI	RW		Bandwidth select I value.	0x0
14:10	SELP	RW		Bandwidth select P value.	0x0
15	BYPASSPLL	RW		Bypass PLL input clock is sent directly to the PLL output (default).	0x0
			1	PLL input clock is sent directly to the PLL output.	
			0	use PLL.	
16	BYPASSPOSTDIV2	RW		Bypass of the divide-by-2 divider in the post-divider.	0x0
			1	Bypass of the divide-by-2 divider in the post-divider.	
			0	Use the divide-by-2 divider in the post-divider.	
17	LIMUPOFF	RW		limup_off = 1 in spread spectrum and fractional PLL applications.	0x0
18	BWDIRECT	RW		Control of the bandwidth of the PLL.	0x0
			1	Modify the bandwidth of the PLL directly.	
			0	The bandwidth is changed synchronously with the feedback divider.	
19	BYPASSPREDIV	RW		Bypass of the pre-divider.	0x0
			1	Bypass of the pre-divider.	
			0	Use the pre-divider.	
20	BYPASSPOSTDIV	RW		Bypass of the post-divider.	0x0
			1	Bypass of the post-divider.	
			0	Use the post-divider.	
21	CLKEN	RW		Enable the output clock.	0x0
			1	Enable the output clock.	
			0	Disable the output clock.	
22	FRMEN	RW		1: free running mode.	0x0
23	FRMCLKSTABLE	RW		Free running mode clockstable: Warning: Only make frm_clockstable = 1 after the PLL output frequency is stable.	0x0
24	SKEWEN	RW		Skew mode.	0x0
			1	Skewmode is enable.	
			0	Skewmode is disable.	
31:25		WO		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.69.2.2 PLL1 status register

The read-only PLL1STAT register provides the PLL lock status and other status details.

Remark: The lock status does not reliably indicate the PLL status for the following two configurations: spread-spectrum mode or fractional enabled or low input clock frequencies such as 32 kHz. In these cases, refer to the PLL lock times listed in the specific device data sheet to obtain appropriate wait times for the PLL to lock.

Table 131. PLL1 status register (PLL1STAT, offset = 0x564)

Bit	Symbol	Access	Description	Reset value
0	LOCK	RO	Lock detector output (active high) Warning: The lock signal is only reliable between fref[2] :100 kHz to 20 MHz.	0x0
1	PREDIVACK	RO	Pre-divider ratio change acknowledge.	0x0
2	FEEDDIVACK	RO	Feedback divider ratio change acknowledge.	0x0
3	POSTDIVACK	RO	Post-divider ratio change acknowledge.	0x0
4	FRMDET	RO	Free running detector output (active high).	0x0
31:5		WO	Reserved. Read value is undefined, only zero should be written.	undefined

4.5.69.2.3 PLL1 N-divider register

The PLL1NDEC controls operation of the PLL pre-divider.

Table 132. PLL1 N divider (PLL1NDEC, offset = 0x568)

Bit	Symbol	Access	Description	Reset value
7:0	NDIV	RW	Pre-divider, divider ratio (N-divider).	0x0
8	NREQ	RW	Pre-divider ratio change request.	0x0
31:9	-	WO	Reserved. Read value is undefined, only zero should be written.	undefined

4.5.69.2.4 PLL1 M-divider register

The PLL1MDEC controls operation of the PLL feedback divider.

Table 133. PLL1 M divider (PLL1MDEC, offset = 0x56C)

Bit	Symbol	Access	Description	Reset value
15:0	MDIV	RW	Feedback divider, divider ratio (M-divider).	0x0
16	MREQ	RW	Feedback ratio change request.	0x0
31:17	-	WO	Reserved. Read value is undefined, only zero should be written.	undefined

4.5.69.2.5 PLL1 P-divider register

The PLL1PDEC controls operation of the PLL post-divider.

Table 134. PLL1 P divider (PLL1PDEC, offset = 0x570)

Bit	Symbol	Access	Description	Reset value
4:0	PDIV	RW	Feedback divider, divider ratio (M-divider).	0x0
5	PREQ	RW	Feedback ratio change request.	0x0
31:6	-	WO	Reserved. Read value is undefined, only zero should be written.	undefined

4.5.70 Functional retention control register (FUNCRETENTIONCTRL)

Disclaimer: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

This register describes the Functional retention control.

Table 135. Functional retention control (FUNCRETENTIONCTRL, offset = 0x704)

Bit	Symbol	Access	Value	Description	Reset value
0	FUNCRETENA	RW		Functional retention in power down only.	0x0
			1	Enable functional retention.	
			0	Disable functional retention.	
13:1	RET_START	RW		Start address divided by 4 inside SRAMX bank.	0x0
23:14	RET_LENTH	RW		Length of Scan chains to save.	0x143
31:24	-	W		Reserved. Read value is undefined, only zero should be written.	undefined.

4.5.71 CPU control for multiple processor

The CPUCTRL register provides control for the two CPUs. CPU0 is factory set to be the master and comes out of reset default. CPU0 cannot be reset or have its clock disabled via this register. Only CPU0 can use the Power APIs, see Chapter 14 "LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API" to cause the device to enter reduced power modes.

If the clock to the CPU1 is to be disabled at some point in the application for power savings, CPU1 should have entered its own sleep mode prior to that point. It avoids incomplete operations in CPU1. CPU1 is in reset and stays disabled until CPU0 decides to release it from reset. Before releasing reset of CPU1, CPU0 should program a set of registers to boot location. This is achieved by programming the CPUCTRL and CPBOOT registers. CPU1 has access to all interrupts as CPU0.

Table 136. CPU Control for multiple processors (CPUCTRL, offset = 0x800)

Bit	Symbol	Access	Value	Description	Reset value
2:0	-	WO		Reserved. Read value is undefined, only zero should be written.	undefined.
3	CPU1CLKEN	RW		CPU1 clock enable.	0x1
			0	The CPU1 clock is enabled.	
			1	The CPU1 clock is not enabled.	
4	-			Reserved. Read value is undefined, only zero should be written.	undefined.
5	CPU1RSTEN	RW		CPU1 reset.	0x1
			0	The CPU1 is being reset.	
			1	The CPU1 is not being reset.	
15:6	-	WO	-	Reserved. Read value is undefined, only zero should be written.	-
31:16		RW		Must be written as 0xC0C4 for the write to have an effect.	-

4.5.72 CPU1 boot address

CPBOOT can be used in an application that uses both CPUs to send CPU1 to an appropriate boot address. CPBOOT sets CPU1 *Vector Table Offset Register* (128 byte aligned). When CPU1 starts execution, its stack pointer and execution addresses are loaded from this table. The CPU1 stack pointer is the first entry and the execution address is in the second entry.

Table 137. Coprocessor boot address (CPBOOT, offset = 0x804)

Bit	Symbol	Description	Reset value
31:0	CPBOOT	Coprocessor boot address.	0x0

4.5.73 CPU status

CPU_STAT provides some status for both CPUs. This register can be read by software at run time, or with a debugger.

Table 138. CPU status (CPSTAT, offset = 0x80C)

Bit	Symbol	Value	Description	Reset value
0	CPU0SLEEPING		The CPU0 sleeping state.	0x0
		1	The CPU is sleeping.	
		0	The CPU is not sleeping.	
1	CPU1SLEEPING		The CPU1 sleeping state.	0x0
		1	The CPU is sleeping.	
		0	The CPU is not sleeping.	
2	CPU0LOCKUP		The CPU0 lockup state.	0x0
		1	The CPU is in lockup.	
		0	The CPU is not in lockup.	
3	CPU1LOCKUP		The CPU1 lockup state.	0x0
		1	The CPU is in lockup.	
		0	The CPU is not in lockup.	
31:4	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.74 DICE register

LPC55S6x offers support for Device Identifier Composition Engine (DICE) to provide Composite Device Identifier (CDI). 8x 32-bit R/W DICE registers are available to store CDI computed by ROM. Once CDI is computed and consumed, contents of those registers will be erased by ROM. Those registers can be used as scratch registers.

Table 139. (From DICE_REG0 to DICE_REG7, offset = 0x900 to offset = 0x91C) bit description

Bit	Symbol	Value	Description	Reset value
31:0	DICE_REGx		Can be used as scratch register.	0x0

4.5.75 Clock control

This register disables clock distribution to prevent extra consumption when they are unused.

Table 140. Various system clock controls: (CLOCK_CTRL, offset = 0xA18)

Bit	Symbol	Access	Value	Description	Reset value
0	-	RW		Reserved.	undefined
1	XTAL32MHZ_FREQM_ENA	RW		Enable XTAL32MHz clock for Frequency Measure module.	0x0
			1	The clock is enabled.	
			0	The clock is not enabled.	

Table 140. Various system clock controls: (CLOCK_CTRL, offset = 0xA18) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
2	FRO1MHZ_UTICK_ENA	RW		Enable FRO 1 MHz clock for Micro-tick timer module.	0x0
			1	The clock is enabled.	
			0	The clock is not enabled.	
3	FRO12MHZ_FREQM_ENA	RW		Enable FRO 12 MHz clock for frequency measure module.	0x0
			1	The clock is enabled.	
			0	The clock is not enabled.	
4	FRO_HF_FREQM_ENA	RW		Enable FRO_HF clock for Frequency Measure module.	0x0
			1	The clock is enabled.	
			0	The clock is not enabled.	
5	CLKIN_ENA	RW		Enable CLCKIN from XTAL32M clock for clock module.	0x0
			1	The clock is enabled.	
			0	The clock is not enabled.	
6	FRO1MHZ_CLK_ENA	RW		Enable 1 MHz FRO.	0x0
			1	The clock is enabled.	
			0	The clock is not enabled.	
7	ANA_FRO12M_CLK_ENA	RW		Enable FRO 12 MHz clock for analog control of the FRO 192 MHz.	0x0
			1	The clock is enabled.	
			0	The clock is not enabled.	
8	XO_CAL_CLK_ENA	RW		Enable clock for both crystal oscillator calibration.	0x0
			1	The clock is enabled.	
			0	The clock is not enabled.	
9	PLU_DEGLITCH_CLK_ENA	RW		Enable clocks FRO_1 MHz and FRO_12 MHz for PLU glitch removal.	0x0
			1	The clock is enabled.	
			0	The clock is not enabled.	
31:10		WO		Reserved. Read value is undefined, only zero should be written.	0x0

4.5.76 Comparator interrupt control

This register is to control the interrupt handler for comparator.

Table 141. Comparator interrupt control (COMP_INT_CTRL, offset = 0xB10)

Bit	Symbol	Value	Description	Reset value
0	INT_ENABLE		Analog comparator interrupt enable control.	0x0
		1	Interrupt enable.	
		0	Interrupt disable.	

Table 141. Comparator interrupt control (COMP_INT_CTRL, offset = 0xB10) *…continued*

Bit	Symbol	Value	Description	Reset value
1	INT_CLEAR		Analog comparator interrupt clear.	0x0
		0	No effect.	
		1	Clear the interrupt. Self-cleared bit.	
4:2	INT_CTRL		Comparator interrupt type selector.	0x0
		0	Analog comparator interrupt edge sensitive is disabled.	
		2	Analog comparator interrupt is rising edge sensitive.	
		4	Analog comparator interrupt is falling edge sensitive.	
		6	Analog comparator interrupt is rising and falling edge sensitive.	
		1	Analog comparator interrupt level sensitive is disabled.	
		3	Analog comparator interrupt is high level sensitive.	
		5	Analog comparator interrupt is low level sensitive.	
		7	Analog comparator interrupt level sensitive is disabled.	
5	INT_SOURCE		Select which analog comparator output (filtered our un-filtered) is used for interrupt detection.	0x0
		0	Select analog comparator filtered output as input for interrupt detection.	
		1	Select analog comparator raw output (unfiltered) as input for interrupt detection should be used when analog comparator is used as wake up source in power down mode.	
31:6	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.77 Comparator interrupt status

This register indicates comparator interrupt status.

Table 142. Comparator interrupt status (COMP_INT_STATUS, offset = 0xB14)

Bit	Symbol	Value	Description	Reset value
0	STATUS		Interrupt status BEFORE interrupt enable.	0x0
		0	No interrupt pending.	
		1	Interrupt pending.	
1	INT_STATUS		Interrupt status AFTER interrupt enable.	0x0
		0	No interrupt pending.	
		1	Interrupt pending.	
2	VAL		Comparator analog output.	0x0
		1	P+ is greater than P-.	
		0	P+ is smaller than P-.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.78 Control automatic clock gating

This register allows selective enabling of automatic clock gating for some peripherals (see Table below). Enabling automatic clock gating will turn off clocks to each peripheral after 16 bus clocks with no activity. This saves power when the peripherals are not used for

some time. When peripherals are turned off because of automatic clock gating, there is a 1 clock delay for the next access. For time-critical code, Automatic clock gating may be disabled to improve speed by 1 to 2%.

Table 143. Control automatic clock gating (AUTOCLKGATEOVERRIDE, offset = 0xE04)

Bit	Symbol	Value	Description	Reset value
0	ROM		Control automatic clock gating of ROM controller.	0x1
		1	Automatic clock gating is disabled.	
		0	Automatic clock gating is enabled	
1	RAMX_CTRL		Control automatic clock gating of RAMX controller.	0x1
		1	Automatic clock gating is disabled.	
		0	Automatic clock gating is enabled	
2	RAM0_CTRL		Control automatic clock gating of RAM 0 controller.	0x1
		1	Automatic clock gating is disabled.	
		0	Automatic clock gating is enabled	
3	RAM1_CTRL		Control automatic clock gating of RAM 1 controller.	0x1
		1	Automatic clock gating is disabled.	
		0	Automatic clock gating is enabled	
4	RAM2_CTRL		Control automatic clock gating of RAM 2 controller.	0x1
		1	Automatic clock gating is disabled.	
		0	Automatic clock gating is enabled	
5	RAM3_CTRL		Control automatic clock gating of RAM 3 controller.	0x1
		1	Automatic clock gating is disabled.	
		0	Automatic clock gating is enabled	
6	RAM4_CTRL		Control automatic clock gating of RAM 4 controller.	0x1
		1	Automatic clock gating is disabled.	
		0	Automatic clock gating is enabled	
7	SYNC0_APB		Control automatic clock gating of synchronous bridge controller 0.	0x1
		1	Automatic clock gating is disabled.	
		0	Automatic clock gating is enabled	
8	SYNC1_APB		Control automatic clock gating of synchronous bridge controller 1.	0x1
		1	Automatic clock gating is disabled.	
		0	Automatic clock gating is enabled	
10:9	-	-	Reserved. Must be written with 1. Ignored upon READ.	
11	CRCGEN		Control automatic clock gating of CRCGEN controller.	0x1
		1	Automatic clock gating is disabled.	
		0	Automatic clock gating is enabled	
12	SDMA0		Control automatic clock gating of DMA0 controller.	0x1
		1	Automatic clock gating is disabled.	
		0	Automatic clock gating is enabled	
13	SDMA1		Control automatic clock gating of DMA1 controller.	0x1
		1	Automatic clock gating is disabled.	
		0	Automatic clock gating is enabled	

Table 143. Control automatic clock gating (AUTOCLKGATEOVERRIDE, offset = 0xE04) *…continued*

Bit	Symbol	Value	Description	Reset value
14	USB0		Control automatic clock gating of USB0 controller (USB Full Speed).	0x1
		1	Automatic clock gating is disabled.	
		0	Automatic clock gating is enabled	
15	SYSCON		Control automatic clock gating of synchronous system controller registers bank.	0x1
		1	Automatic clock gating is disabled.	
		0	Automatic clock gating is enabled	
31:16	ENABLEUPDATE		The value 0xC0DE must be written for AUTOCLKGATEOVERRIDE registers fields updates to have effect.	0x0
		0xC0DE	Bit Fields 0 - 15 of this register are updated.	
		-	Any other values than 0xC0DE. Bit Fields 0 - 15 of this register are not updated.	

4.5.79 Enable bypass of the first stage

This register enable bypass of the first stage of synchronization inside GPIO_INT module.

Table 144. Control of synchronization inside GPIO_INT module (GPIOPSYNC, offset = 0xE08)

Bit	Symbol	Value	Description	Reset value
0	PSYNC		Enable bypass of the first stage of synchronization inside GPIO_INT modules.	0x0
		1	Bypass of the first stage of synchronization inside GPIO0, GPIO1, GINT0, GINT1, PINT0 and PINT1 modules.	
		0	Use the first stage of synchronization inside GPIO0, GPIO1, GINT0, GINT1, PINT0 and PINT1 modules.	
31:1	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.80 Debug lock enable

This register controls write access to CODESECURITYPROTTEST, CODESECURITYPROTCPU0, CODESECURITYPROTCPU1, CM33_DEBUG_FEATURES, MCM33_DEBUG_FEATURES and DBG_AUTH_SCRATCH registers.

Table 145. Debug Lock Enable (DEBUG_LOCK_EN, offset = 0xFA0)

Bit	Symbol	Value	Description	Reset value
3:0	LOCK_ALL		Control write access to security registers: Control write access to CODESECURITYPROTTEST, CODESECURITYPROTCPU0, CODESECURITYPROTCPU1, CPU0_DEBUG_FEATURES, CPU1_DEBUG_FEATURES and DBG_AUTH_SCRATCH registers.	0xA[1]
		0xA	Enables write access to all six registers.	
		0x5	Disables write access to all six registers. Once 0x5 is written in this field, its value cannot be modified until a Power On Reset (PoR) occurs.	
		Others	Reserved.	
31:4	-		Reserved. Read value is undefined, only zero should be written.	undefined

[1] Upon POR, ROM will configure the security related registers based on PFR and will write 0x5 to DEBUG_LOCK_EN register.

4.5.81 Debug features control

This register controls CPU0 and CPU1 debug features. Invasive debug is defined as a debug process where you can control and observe the processor like halting processor and modifying its state.

Table 146. Debug Features register (DEBUG_FEATURES, offset = 0xFA4)

Bit	Symbol	Value	Description	Reset value
1:0	CPU0_DBGEN		CPU0 invasive debug control.	0x0
		0x2	Invasive debug is enabled.	
		0x1	Invasive debug is disabled.	
		Others	Reserved.	
3:2	CPU0_NIDEN		CPU0 non invasive debug control.	0x0
		0x2	Invasive debug is enabled.	
		0x1	Invasive debug is disabled.	
		Others	Reserved.	

Table 146. Debug Features register (DEBUG_FEATURES, offset = 0xFA4) *…continued*

Bit	Symbol	Value	Description	Reset value
5:4	CPU0_SPIDEN		CPU0 secure invasive debug control.	0x0
		0x2	Invasive debug is enabled.	
		0x1	Invasive debug is disabled.	
		Others	Reserved.	
7:6	CPU0_SPNIDEN		CPU0 secure non invasive debug control.	0x0
		0x2	Invasive debug is enabled.	
		0x1	Invasive debug is disabled.	
		Others	Reserved.	
9:8	MCM33_DBGEN		CPU1 invasive debug control.	0x0
		0x2	Invasive debug is enabled.	
		0x1	Invasive debug is disabled.	
		Others	Reserved.	
11:10	MCM33_NIDEN		CPU1 non invasive debug control.	0x0
		0x2	Invasive debug is enabled.	
		0x1	Invasive debug is disabled.	
		Others	Reserved.	
31:12	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.82 Debug features control duplicate

This register controls the CPU0 and CPU1 debug features. It is a duplicate of the Debug Features register. This duplicated register with multi-bit control is provided to counter fault attacks.

Table 147. Debug Features Duplicate register (DEBUG_FEATURES_DP, offset = 0xFA8)

Bit	Symbol	Value	Description	Reset value
1:0	CPU0_DBGEN		CPU0 Invasive debug control.	0x1
		0x2	Invasive debug is enabled.	
		0x1	Invasive debug is disabled.	
		Others	Reserved.	
3:2	CPU0_NIDEN		CPU0 non invasive debug control.	0x1
		0x2	Invasive debug is enabled.	
		0x1	Invasive debug is disabled.	
		Others	Reserved.	
5:4	CPU0_SPIDEN		CPU0 secure invasive debug control.	0x1
		0x2	Invasive debug is enabled.	
		0x1	Invasive debug is disabled.	
		Others	Reserved.	
7:6	CPU0_SPNIDEN		CPU0 secure non invasive debug control.	0x1
		0x2	Invasive debug is enabled.	
		0x1	Invasive debug is disabled.	
		Others	Reserved.	

Table 147. Debug Features Duplicate register (DEBUG_FEATURES_DP, offset = 0xFA8) *…continued*

Bit	Symbol	Value	Description	Reset value
9:8	MCM33_DBGEN		CPU1 invasive debug control.	0x1
		0x2	Invasive debug is enabled.	
		0x1	Invasive debug is disabled.	
		Others	Reserved.	
11:10	MCM33_NIDEN		CPU1 non invasive debug control.	0x1
		0x2	Invasive debug is enabled.	
		0x1	Invasive debug is disabled.	
		Others	Reserved.	
31:12	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.83 SWD access port for CPU0

This register is used by ROM during DEBUG authentication mechanism to enable debug access port for CPU0.

Table 148. SWD access port for CPU0 (SWDCPU0, offset = 0xFB4)

Bit	Symbol	Value	Description	Reset value
31:0	SEC_CODE		CPU0 SWD-AP: 0x12345678.	0x0
		305419896	Value to write to enable CPU0 SWD access. Reading back register will be read as 0xA.	
		Others	CPU0 SWD is not allowed. Reading back register will be read as 0x5.	

4.5.84 SWD access port for CPU1

This register is used by ROM during DEBUG authentication mechanism to enable debug access port for CPU1.

Table 149. SWD access port for CPU1 (SWDCPU1, offset = 0xFB8)

Bit	Symbol	Value	Description	Reset value
31:0	SEC_CODE		CPU1 SWD-AP: 0x12345678.	0x0
		305419896	Value to write to enable CPU1 SWD access. Reading back register will be read as 0xA.	
		Others	CPU1 SWD is not allowed. Reading back register will be read as 0x5.	

4.5.85 Key block register

Write a value in this register to block access to PUF indexes. This register is used to detect tamper attacks. Any value other than 0x3CC35AA5 written to this register will disable PUF output. Once disabled, keys cannot be retrieved from PUF.

Table 150. Key Block register (KEY_BLOCK, offset = 0xFBC)

Bit	Symbol	Value	Description	Reset value
31:0	KEY_BLOCK		Write a value to block PUF indexes.	0x3CC35AA5

4.5.86 Debug authentication BEACON register

This register is protected by security. ROM sets this register (read only) with a value received in the debug credentials before passing control to user code. This is useful for extending debug authentication control for customer applications. Please refer to Section 51.7 “Debug authentication” for details.

Table 151. Debug Authentication Scratch registers (DEBUG_AUTH_BEACON, offset = 0xFC0)

Bit	Symbol	Description	Reset value
31:0	BEACON	Set by the debug authentication code in ROM to pass the debug beacons (Credential Beacon and Authentication Beacon) to application code.	0x0

4.5.87 CPU configuration register

This register is used to configure CPU1.

Table 152. CPU Configuration register (CPUCFG, offset = 0xFD4)

Bit	Symbol	Value	Description	Reset value
1:0	-		Reserved.	0x2
2	CPU1ENABLE		Enable CPU1.	0x0
		0	CPU1 is disabled (Processor in reset).	
		1	CPU1 is enabled.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined.

4.5.88 Device ID register

This register describes the device ID.

Table 153. Device ID0 register (DEVICE_ID0, offset = 0xFF8)

Bit	Symbol	Value	Description	Reset value
31:0	DEVICE_ID		Device/Part Identification	
		0xA010111C	LPC5528 Rev 1B	undefined.
		0xA010221A	LPC5526 Rev 1B	undefined.
		0xA010229A	LPC55S26 Rev 1B	undefined.
		0xA010119C	LPC55S28 Rev 1B	undefined.
		0x501022C2	LPC55S66 Rev 1B	undefined.
		0x501000C5	LPC55S69 Rev 1B	undefined.
		0x5000220	LPC55S66 Rev 0A	undefined.
		0x5000000	LPC55S69 Rev 0A	undefined.

4.5.89 Chip revision ID and N number

This register describes the Chip Number and Revision.

Table 154. Chip revision ID and number (DIEID, offset = 0xFFC)

Bit	Symbol	Description	Reset value
3:0	REV_ID	Device Revision. Value read as 0x0 applies to device revision 0A and 0x1 to device revision 1B.	0x1
31:4	-	Reserved. Read value is undefined, only zero should be written.	undefined.

4.6 Functional description

4.6.1 Reset

Reset has the following sources:

- The $\overline{\text{RESET}}$ pin.
- Watchdog reset.
- Power-On Reset (POR).
- Brown Out Detect (BOD).
- ARM system reset.
- ISP-AP debug reset.
- Software reset.

Assertion of the POR or the BOD reset, once the operating voltage attains a usable level, starts the FRO_192. After the FRO-start-up time, the FRO_192 provides a stable clock output. The reset remains asserted until the external reset is released, the oscillator is running, and the flash controller has completed its initialization.

On the assertion of any reset source (ARM system reset, POR, BOD reset, external reset, watchdog reset, and Software Reset), the following processes are initiated:

1. The FRO is enabled or starts up if not running.
2. The flash wake-up starts. This takes approximately 40μs.
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.

4.6.2 Clock

The main clock select multiplexers are implemented with glitch-free logic. All the other clock multiplexers described in this chapter cannot be considered as glitch-free, thus it is necessary to pay attention during clock switching. All the dividers can be halted and restarted during clock switching, to provide a glitch free output.

4.6.3 Start-up behavior

The FRO 12 MHz oscillator provides the default clock at reset and provides a clean system clock shortly after the supply pins reach operating voltage. See the device data sheet for details of start-up timing.

Note: The ROM boot code might switch to a higher frequency (either 24 MHz, 48 MHz, or 96 MHz) based on the settings in the Flash Protected Area (FPR).

4.6.4 Brown-out detection

This device includes one Brown-out detector to monitor the voltage of VBAT. If the voltage falls below one of the selected voltages, see Section 13.4.6 “VBAT Brown Out Detector (BoD) control register” the BOD asserts an interrupt to the NVIC or issues a reset, see Section 13.4.1 “Power Management Controller FSM (Finite State Machines) status (STATUS)”.

The interrupt signal can be enabled for interrupt in the interrupt enable register in the NVIC, see Table 8 to cause a CPU interrupt; if not, software can monitor the signal by reading a dedicated status register.

If the BOD interrupt is enabled, the BOD interrupt can wake up the chip from a reduced power mode, not including power-down and deep power-down. See Chapter 14 “LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API”.

If the BOD reset is enabled, the forced BOD reset can wake up the chip from reduced power modes, not including power-down and deep power-down.

4.6.5 Flash accelerator functional description

The flash accelerator is also known as the Flash Memory Controller, or FMC. The FMC is distinct from, and interfaces with the Flash Controller.

The flash accelerator block maximizes performance of the CPU, (when it is running code from flash memory), and helps to reduce power consumption.

See Section 4.5.61 “FMC configuration register” for more details.

The flash accelerator is divided into several functional blocks:

- AHB matrix interface, accessible by all bus masters that have a connection to the matrix slave port used for flash memory.
- An array of eight 128-bit buffers.
- Flash accelerator control logic, including address compare and flash control.
- A flash memory interface.

Figure 7 shows a simplified diagram of the flash accelerator blocks and data paths.

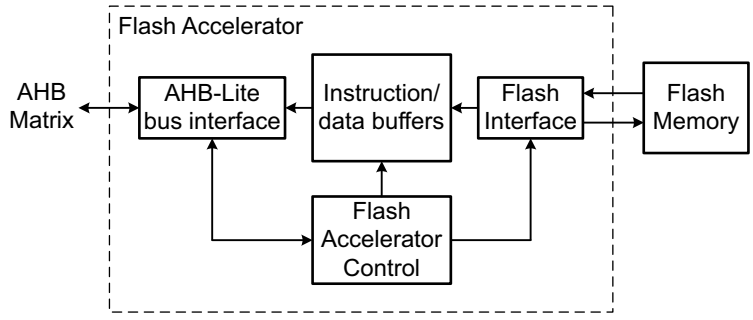


Fig 7. Simplified block diagram of the flash accelerator

Note: In the following descriptions, the term *"fetch"* applies to an explicit flash read request from the CPU.

4.6.5.1 Flash memory bank

Flash programming operations are not controlled by the flash accelerator, but are handled as a separate function. The boot code includes flash programming functions that may be called as part of the application program, as well as loaders that may be used to accomplish initial flash programming.

4.6.5.2 Flash programming constraints

Since the flash memory does not allow accesses during programming and erase operations, it is necessary for the flash accelerator to force the CPU to wait if a memory access to a flash address is requested while the flash memory is busy with a programming operation. Under some conditions, this delay could result in a Watchdog time-out. The user will need to be aware of this possibility and take steps to insure that an unwanted Watchdog reset does not cause a system failure while programming or erasing the flash memory. Application code, especially interrupts, can continue to run from other memories during flash erase/write operations.

In order to preclude the possibility of stale data being read from the flash memory, the flash accelerator buffers are automatically invalidated at the beginning of any flash programming or erase operation. Any subsequent read from a flash address will cause a new fetch to be initiated after the flash operation has completed.

Note: Flash ERASE and PROGRAM operations must be performed with a system clock below or equal to 100 MHz.

4.6.6 PLL0 and PLL1 functional description

The PLL is typically used to create a frequency that is higher than other on-chip clock sources, and used to operate the CPU and/or other on-chip functions. It may also be used to obtain a specific clock that is otherwise not available. For example, a source clock with a frequency of any integer MHz (for example, the 12 MHz FRO) can be divided down to 1 MHz, then multiplied up to any other integer MHz (for example, 13, 14 and15).The PLL can be set up by calling an API supplied by NXP Semiconductors. Also see Section 4.5.69 "PLL registers", and Section 14.4.2 "POWER_EnterSleep".

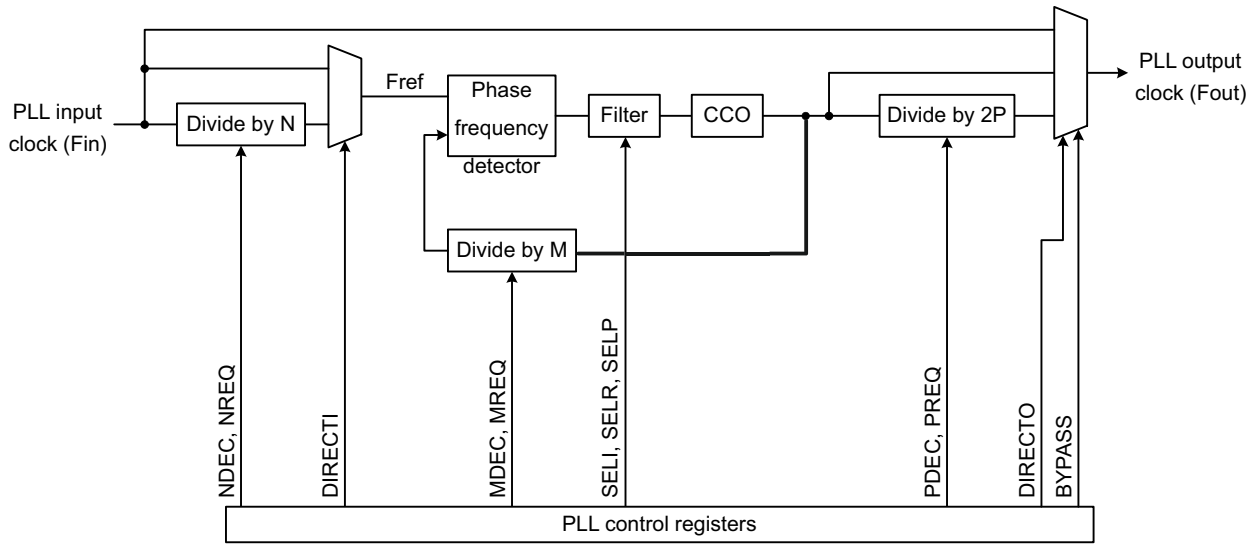


Fig 8. PLL block diagram showing typical operation

4.6.6.1 PLL features

- Integrated PLL with no external components for clock generation.
- Large input range at the phase detector: 2 kHz - 150 MHz.
- CCO frequency: 275 MHz - 550 MHz.
- Output clock (clkout) range: 4.3 MHz to 550 MHz (max limited to 150 MHz).
- Programmable:
  - Pre-divider N, (N, 1 to 2^8-1)
  - Feedback-divider M, (M, 1 to 2^{16}-1)
  - Post-divider P * 2 (P, 1 to 2^5-1)
- Programmable bandwidth (integrating action, proportional action, high frequency pole).
- Real-time adjustment of the clock (dividers with handshake control).
- Positive edge clocking.
- Frequency limiter to avoid *hang-up* of the PLL.
- Lock detector.
- Power-down mode.
- Possibility to bypass whole PLL.
- Possibility to bypass the post-divider.
- Possibility to bypass the pre-divider.
- Possibility to disable the output clock.
- Spread Spectrum mode (only on PLL0).

4.6.6.2 PLL description

A number of sources may be used as an input to the PLL, see Figure 4 “LPC55S6x Clock generation (Part 1 of 3)”. In addition, a block diagram of the PLL is shown in Figure 8 “PLL block diagram showing typical operation”. The PLL input, in the range: 2 kHz to 150 MHz, may initially be divided down by a value *N*, which may be in the range of 1 to 255. This input division provides a greater number of possibilities in providing a wide range of output frequencies from the same input frequency.

Following the PLL input divider is the PLL multiplier. The multiplier can multiply the input divider output through the use of a Current Controlled Oscillator (CCO) by a value *M*, in the range of 1 through 65,535. The resulting frequency must be in the range of 275 MHz to 550 MHz. The multiplier works by dividing the CCO output by the value of M, then using a phase-frequency detector to compare the divided CCO output to the multiplier input. The error value is filtered and used to adjust the CCO frequency.

The PLL output may further be divided by a value *2P* if desired, where P is value in the range of 1 to 31.

All of the dividers that are part of the PLL use an encoded value, not the binary divide value. The LPCOpen Chip_POWER_SetPLL API, see Section 14.4.2 “POWER_EnterSleep” can adjust the value for the main feedback divider (the M divider), but does not accept pre- and post-divider values. See section Section 4.6.6.3 “PLL operating modes” and Section 4.6.6.5 “PLL usage” for information on how to obtain divider values.

There are additional dividers in the clocking system to bring the PLL output frequency down to what is needed for the CPU, USB, and other peripherals. The PLL output dividers are described in the Clock Dividers section following the PLL description.

For PLL register descriptions, see Section 4.5.69 “PLL registers”.

4.6.6.2.1 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 seven 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 seven 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.

The PLL lock indicator is not reliable when F_{ref} is below 100 kHz or above 20 MHz. Instead, software should use a 6 ms time interval to insure the PLL will be stable.

For PLL0, spread spectrum mode, the PLL will generally not lock, software should use a 6 ms time interval to insure the PLL will be stable. See Section 4.6.6.5.1 “Procedure for determining PLL settings”.

Remark: Please see PLL Lock Bit errata regarding details on ensuring the PLL is stable and locked.

4.6.6.2.2 Power-down

To reduce the power consumption when the PLL clock is not needed, a PLL power-down mode has been incorporated. This mode is enabled by setting the PDEN_PLLn (where n indicates PLL number) bit to one in the power configuration register PDRUNCFG0, see Table 311. In this mode, the internal current reference will be turned off, the oscillator and the phase-frequency detector will be stopped and the dividers will enter a reset state. While in PLL power-down mode, the lock output will be low to indicate that the PLL is not in lock.

When the PLL power-down mode is terminated by setting the PDEN_PLLn (where n indicates PLL number) 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. While in this state, new divider values may be entered, which will be used when the PLL power-down state is exited by clearing PDEN_PLLn (where n indicates PLL number).

4.6.6.3 PLL operating modes

The PLL includes several main operating modes, and a power-down mode. These are summarized in Table 155 and detailed in the following sections.

Table 155. PLL operating mode summary

Mode	PDEN_PLLn (where n indicates PLL number) bit in PDRUNCFG0	Bits in SYSPLLCTRL:			SEL_EXT bit in PLL0SSCG0	PD bit in PLL0SSCG1
		BYPASS	UPLIMOFF	BANDSEL		
Normal	0	0	0	1	1	1
Spread spectrum (only for PLL0)	0	0	1	0	0	0
Power-down	1	x [1]	x	x	x	1

[1] Use 1 if the PLL output is used even though the PLL is not altering the frequency.

4.6.6.3.1 Normal modes

Typical operation of the PLL includes an optional pre-divide of the PLL input, followed by a frequency multiplication, and finally an optional post-divide to produce the PLL output.

Notations used in the frequency equations:

- Fin = the input to the PLL.
- Fout = the output of the PLL.
- Fref = the PLL reference frequency, the input to the phase frequency detector.
- N = optional pre-divider value.
- M = feedback divider value, which represents the multiplier for the PLL.
- P = optional post-divider value. An additional divide-by-2 is included in the post-divider path.

A block diagram of the PLL as used in normal modes is shown in Figure 9.

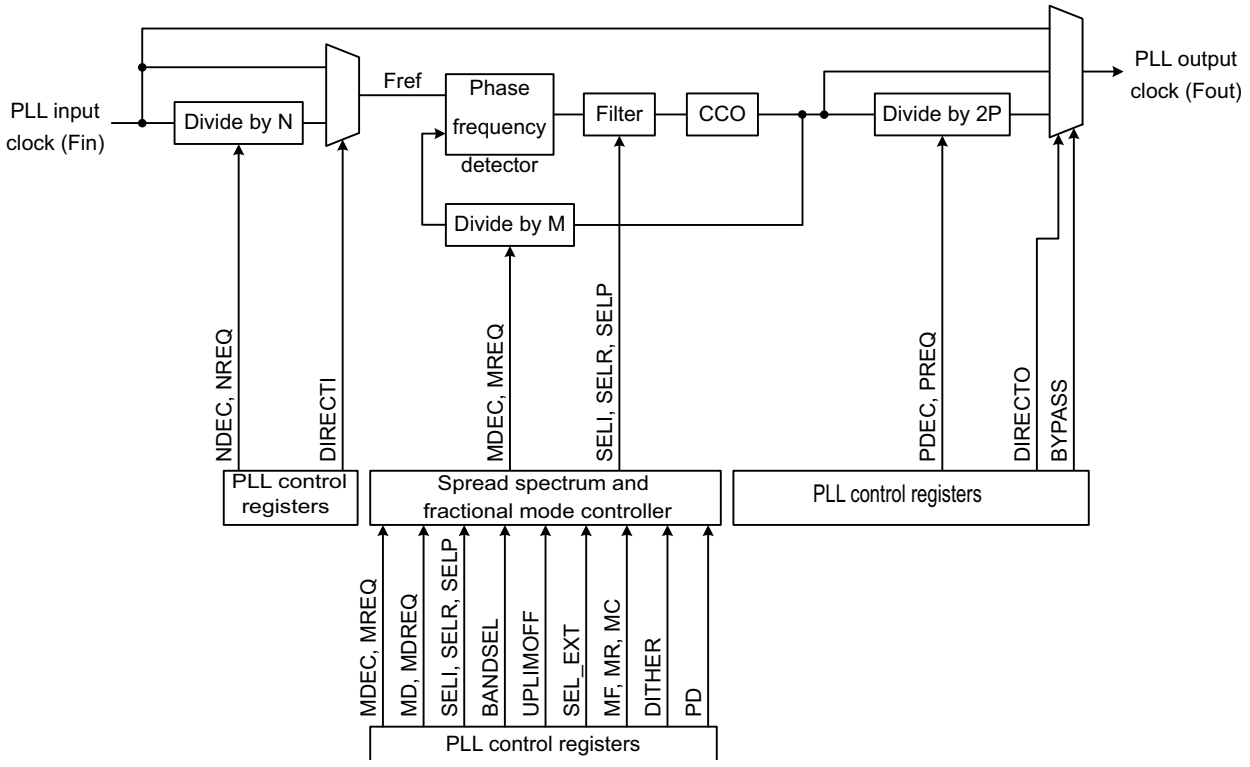


Fig 9. PLL0 block diagram showing spread spectrum and fractional divide operation

Mode 1a: Normal operating mode without post-divider and without pre-divider

In normal operating mode 1a the post-divider and pre-divider are bypassed. The operating frequencies are:

$F_{out} = F_{cco} = M \times F_{in} \wedge (275\ MHz \leq F_{cco} \leq 550\ MHz, 2\ kHz \leq F_{in} \leq 150\ MHz)$

The feedback ratio is programmable:

- Feedback-divider M (M, 1 to $2^{16} - 1$)

Mode 1b: Normal operating mode with post-divider and without pre-divider

In normal operating mode 1b the pre-divider is bypassed. The operating frequencies are:

$F_{out} = F_{cco} / (2 \times P) = M / (2 \times P) \times F_{in} \wedge (275\ MHz \leq F_{cco} \leq 550\ MHz, 2\ kHz \leq F_{in} \leq 150\ MHz)$

The divider ratios are programmable:

- Feedback-divider M (M, 1 to $2^{16} - 1$)
- Post-divider P (P, 1 to $2^{5} - 1$)

Mode 1c: Normal operating mode without post-divider and with pre-divider

In normal operating mode 1c the post-divider with divide-by-2 divider is bypassed. The operating frequencies are:

$F_{out} = F_{cco} = M/N \times F_{in} \wedge (275\ MHz \leq F_{cco} \leq 550\ MHz, 2\ kHz \leq F_{in}/N \leq 150\ MHz)$

The divider ratios are programmable:

- Pre-divider N (N, 1 to 2^8 - 1)
- Feedback-divider M (M, 1 to 2^{16} - 1)

Mode 1d: Normal operating mode with post-divider and with pre-divider

In normal operating mode 1d none of the dividers are bypassed. The operating frequencies are:

$F_{out} = F_{cco}/2xP = M/ (N \times 2 \times P) \times F_{in} \wedge (275\ MHz \leq F_{cco} \leq 550\ MHz, 2\ kHz \leq F_{in}/N \leq 150\ MHz)$

The divider ratios are programmable:

- Pre-divider N (N, 1 to 2^8 - 1)
- Feedback-divider M (M, 1 to 2^{16} - 1)
- Post-divider P (P, 1 to 2^5 - 1)

4.6.6.3.2 Selecting the bandwidth

In normal applications the bandwidth must be calculated manually by using the equations below for seli and selp. In that case the PLL will be automatically stable. In normal applications pin band_direct has to be low (‘0’) in this case the bandwidth is changed together with the M-divider value.

For normal applications the value for selp[4:0] must be calculated using the following equation:

selp = floor(M/4) + 1

Where:

- Feedback-divider M (M, 1 to 2^{16} - 1)
- If selpcalculated >= 31 then selp[4:0] = 31

For normal applications the value for seli[5:0] must be calculated using one of the following equations depending on the value of the feedback divider M:

if (M >= 8000) => seli = 1

if (8000 > M >= 122) => seli = floor(8000/M)

if (122 > M >= 1) => seli = 2 * floor(M/4) + 3

Where:

- Feedback-divider M (M, 1 to 2^{16} - 1)
- If seli >= 63 then seli[5:0] = 63.

For normal applications the value for selr[3:0] must be kept 0.

For frequencies at the phase detector smaller than 50 kHz (Fin/N ≤ 50kHz) please consult NXP.

In some applications, it is preferable to change the bandwidth directly on the PLL. In such an application, Bit BWDIRECT in the PLLxCTRL register must be set high (‘1’).

4.6.6.3.3 Spread spectrum mode

The spread spectrum functionality can be used to modulate the PLL output frequency automatically, in a programmable manner. It can decrease electromagnetic interference (EMI) in an application.

The spread spectrum clock generator can be used in several ways:

- It can encode M-divider values between 1 and 255 to produce the MDEC value used directly by the PLL, saving the need for executing encoding algorithm code, or hard-coding predetermined values into an application.
- It can provide a fractional rate feature to the PLL.
- It can be set up to automatically alter the PLL CCO frequency on an ongoing basis to decrease electromagnetic interference (EMI).

A block diagram of the PLL as used in fractional mode is shown in Figure 9 “PLL0 block diagram showing spread spectrum and fractional divide operation”.

If the spread spectrum mode is enabled, choose N to ensure 3 MHz < Fin/N < 5 MHz. Spread spectrum mode cannot be used when Fin = 32 kHz.

When the modulation (MR) is set to zero, the PLL becomes a fractional PLL.

Triangular wave modulation: For the center spread triangular waveform modulation with a modulation frequency depth δ fmodpk-pk and a modulation frequency fm, the clock cycle displacement and spectral tone reduction ΔP can be calculated. The theoretical maximum clock cycle displacement (peak-to-peak) can be expressed with the following equation below:

if directo$_{PLL}$ = 1:

$$\Delta n_{max;theoretically} = \frac{N_{ss} \times k}{16}$$

if directo$_{PLL}$ = 0, P$_{PLL}$ = 1:

$$\Delta n_{max;theoretically} = \frac{N_{ss} \times k}{32 \times P_{PLL}}$$

In practice, the clock cycle displacement could be larger. So, for safety reasons (buffer overflow) use:

if directo$_{PLL}$ = 1:

$$\Delta n_{max;practically} = \frac{N_{ss} \times k}{8}$$

if directo$_{PLL}$ = 0, P$_{PLL}$ = 1:

$$\Delta n_{max;practically} = \frac{N_{ss} \times k}{16 \times P_{PLL}}$$

The spectral tone reduction/EMI reduction ΔP at F$_{out}$ is approximately:

if directo$_{PLL}$ = 1:

$$\Delta P \approx 10log\frac{N_{ss} \times k}{2}$$

if directo$_{PLL}$ = 0, P$_{PLL}$ = 1:

$$\Delta P \approx 10log\frac{N_{ss} \times k}{4 \times P_{PLL}}$$

See Table 156 for the spectral tone reduction and clock cycle displacement for directo$_{PLL}$ = 0 and P$_{PLL}$= 1.

Table 156. Values for different settings, directo$_{PLL}$ = 0, P$_{PLL}$ = 1

Table values are: ΔP Δn$_{max}$	mf[2:0]=000 N$_{SS}$ = 512		mf[2:0]=001 N$_{SS}$ ≈ 384		mf[2:0]=010 N$_{SS}$ = 256		mf[2:0]=011 N$_{SS}$ = 128		mf[2:0]=100 N$_{SS}$ = 64		mf[2:0]=101 N$_{SS}$ = 32		mf[2:0]=110 N$_{SS}$ ≈ 24		mf[2:0]=111 N$_{SS}$ = 16	
mr[2:0]=000, k≈0	0 dB	0	0 dB	0	0 dB	0	0 dB	0	0 dB	0	0 dB	0	0 dB	0	0 dB	0
mr[2:0]=001, k≈1	21 dB	32	20 dB	24	18 dB	16	15 dB	8	12 dB	4	9 dB	2	8 dB	1.5	6 dB	1
mr[2:0]=010, k≈1.5	23 dB	48	22 dB	32	20 dB	24	17 dB	12	14 dB	6	11 dB	3	10 dB	2.2	8 dB	1.5
mr[2:0]=011, k≈2	24 dB	64	23 dB	48	21 dB	32	18 dB	16	15 dB	8	12 dB	4	11 dB	3	9 dB	2
mr[2:0]=100, k≈3	26 dB	96	25 dB	64	25 dB	48	20 dB	24	17 dB	12	14 dB	6	13 dB	4.5	12 dB	4
mr[2:0]=101, k≈4	27 dB	128	26 dB	96	24 dB	64	21 dB	32	18 dB	16	15 dB	8	14 dB	6	12 dB	4
mr[2:0]=110, k≈6	28 dB	192	28 dB	128	26 dB	96	23 dB	48	20 dB	24	17 dB	12	16 dB	9	14 dB	6
mr[2:0]=111, k≈8	30 dB	256	29 dB	192	27 dB	128	24 dB	64	21 dB	32	18 dB	16	17 dB	12	15 dB	8

4.6.6.3.4 Fractional Divide operation

Please refer to Table 125 “PLL0 spread spectrum wrapper control register 0 (PLL0SSCG0, offset = 0x590)” and Table 126 “PLL0 spread spectrum wrapper control register 1 (PLL0SSCG1, offset = 0x594)” registers to use fractional adjustable mode.

.

4.6.6.3.5 PLL power-down mode

If the PLL is not used, or if it there are cases where it is turned off in a running application, power can be saved by putting the PLL in power-down mode. Before this is done, the CPU and any peripherals that are not meant to be stopped as well, must be running from some other clock source.

4.6.6.4 PLL related registers

The PLL is controlled by registers described elsewhere in this chapter, see Section 4.5.69 “PLL registers”, and summarized below.

Table 157. Summary of PLL related registers

Register	Description	Section
PLLxCTRL	PLL control.	4.5.69.1.1
PLLxSTAT	PLL status.	4.5.69.1.2
PLLxNDEC	PLL pre-divider.	4.5.69.1.3
PLLxPDEC	PLL post-divider.	4.5.69.1.4
PLL0SSCTRL	PLL spread spectrum control 0.	4.5.69.1.5
PLL0SSCTRL1	PLL spread spectrum control 1.	4.5.69.1.5

4.6.6.5 PLL usage

As previously noted, the PLL divider settings used in the PLL registers are not simple binary values, they are encoded as shown in the PLL register descriptions. The divider values and their encoding can be found by calculation using the information in this document. For simple PLL usage with no pre- or post-divide, the LPCOpen. Chip_POWER_SetPLL API can be used, see Section 14.4.2 “POWER_EnterSleep”. Also, a PLL setting calculator can be found on the NXP website. The latter two possibilities are recommended in order to avoid PLL setup issues.

4.6.6.5.1 Procedure for determining PLL settings

In general, PLL configuration values may be found as follows:

1. Identify a desired PLL output frequency. This may depend on a specific interface frequency needed or be based on expected CPU performance requirements, and may be limited by system power availability.
2. Determine which clock source to use as the PLL input. This can be influenced by power or accuracy required, or by the potential to obtain the desired PLL output frequency.
3. Identify PLL settings to obtain the desired output from the selected input. The Fcco frequency must be either the actual desired output frequency, or the desired output frequency times 2 x P, where P is from 2 to 31. The Fcco frequency must also be a multiple of the PLL reference frequency, which is either the PLL input, or the PLL input divided by N, where N is from 2 to 255.
4. There may be several ways to obtain the same PLL output frequency. PLL power depends on Fcco (a lower frequency uses less power) and the divider used. Bypassing the input and/or output divider saves power.
5. Check that the selected settings meet all of the PLL requirements:
   - Fin is in the range of 32 kHz to 100 MHz.
   - Fcco is in the range of 275 MHz to 550 MHz.
   - Fout is in the range of 1.2 MHz to 100 MHz.
   - The pre-divider is either bypassed, or N is in the range of 2 to 255.
   - The post-divider is either bypassed, or P is in the range of 2 to 31.
   - M is in the range of 3 to 65,535.

   Also note that PLL startup time becomes longer as Fref drops below 500 kHz. At 500 kHz and above, startup time is up to 500 microseconds. Below 500 kHz, startup time can be estimated as 200 / Fref, or up to 6.1 milliseconds for Fref = 32 kHz. PLL accuracy and jitter is better with higher values of Fref.

4.6.6.5.2 PLL setup sequence

The following sequence should be followed to initialize and connect the PLL:

1. Make sure that the PLL output is disconnected from any downstream functions. If the PLL was previously being used to clock the CPU, and the CPU Clock Divider is being used, it may be set to speed up operation while the PLL is disconnected.
2. Select a PLL input clock source. See Section 4.5.37 “PLL0 clock source select register”.
3. Set up the PLL dividers and mode settings. See Section 4.5.69 “PLL registers”.

4. Wait for the PLL output to stabilize. The start-up time is 500 µs + 300 / Fref seconds.
5. If the PLL will be used to clock the CPU, change the CPU Clock Divider setting for operation with the PLL, if needed. This must be done before connecting the PLL.
6. Connect the PLL to whichever downstream function with which it is being used. The structure of the clock dividers may be seen on the right of Figure 4.

5.1 General description

This chapter describes the flash controller targeted for the LPC55S6x/LPC55S2x/LPC552x device.

5.2 Features

- Includes analog delay block to manage self-timed read operation.
- Read port designed as an interface to the FMC flash cache.
- APB registers interface (separate clock domain with respect to the read port).
- Auto initialization after reset.
- ECC management, including single bit correction and error correction logging.

5.3 Block diagram

Figure 10 shows a functional block diagram of the controller. Some connections between blocks are not presented for clarity. The actual design hierarchy does not correspond to this diagram: the controller top level instantiates the hard blocks (the flash and the analog delay block), and a block that contains all logic. The logic block is subdivided into sub-block for each of the different clock domains, and an additional block that manages all clocks and resets.

The architecture is built around a sequencer, which transforms complex user and test commands into a sequence of basic memory operations. The sequencer implements a number of commands, for example, to change the content of the memory, check its content, and change the mode of operation.

Flash operations (erase, blank check, program) and reading a single word can only be performed for CPU frequencies of up to 100 MHz. These operations cannot be performed for frequencies above 100 MHz.

Remark: When performing AHB reads of the flash memory content at the just erased locations, a hardware fault will occur. Read operations performed using flash controller commands (See Section 5.6.2 “Command listing (CMD)”) will not cause a hard fault. See AN12949 for Flash Programming Tips for LPC5500 Series.

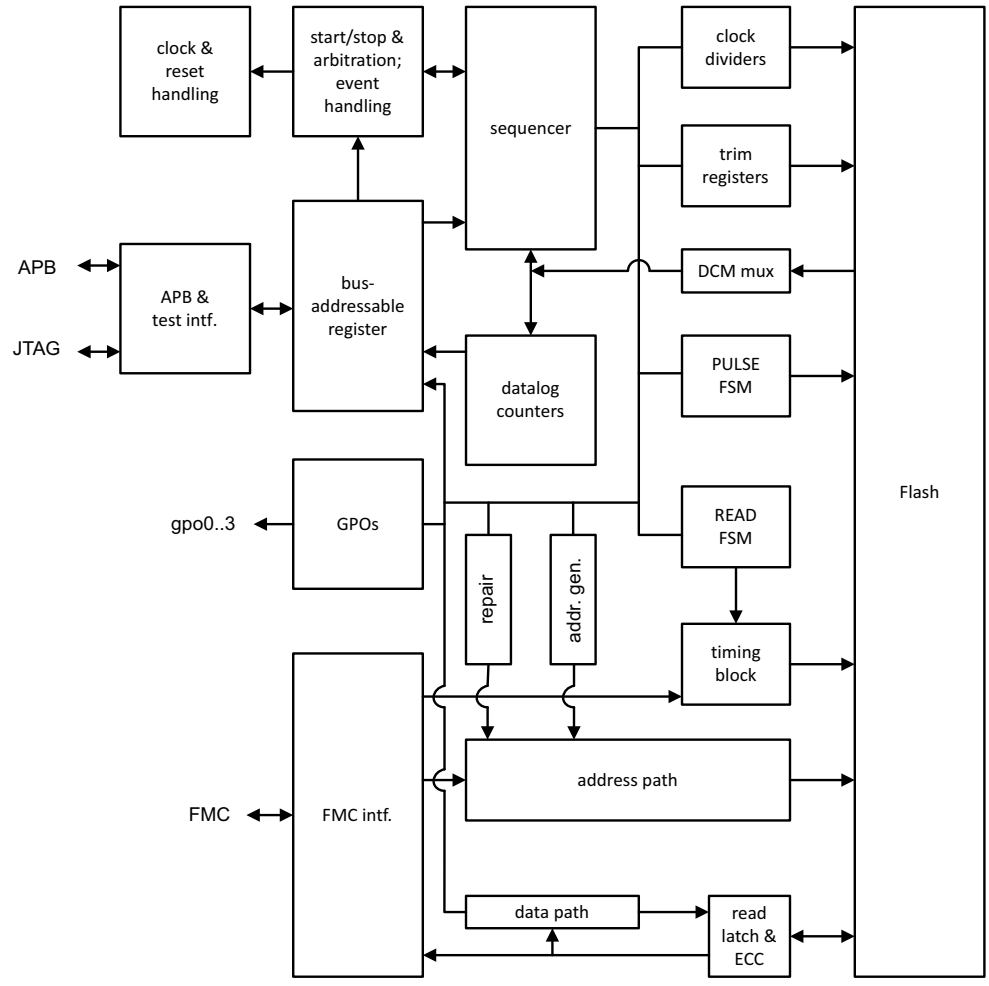


Fig 10. Block diagram

5.4 Software Interface

See Chapter 9 “LPC55S6x/LPC55S2x/LPC552x Flash API” for details.

5.5 Register overview

Control and status information for the controller is mapped into register bits. All registers are 32 bit wide and can only be accessed as a whole word.

See Table 158 for a list of registers. Registers are arranged in an address space which is 4 kbyte wide.

The “Access” field must be interpreted as follows: R = read, W = write, S = set (set asserted bits, leave others unchanged), C = clear (clear asserted bits, leave others unchanged), T= only accessible (R/W) in test mode (reserved in user mode).

S and C are special versions of write access, where the write data does not reflect the new register content, but indicates which bits must be set or cleared.

When multiple access types are supported, multiple characters are given: for example R/W for registers that have both read and write access.

Within an otherwise accessible register, there may be reserved register bits, which can be neither read nor written.When the read access is not specified explicitly, read access is not supported.

Inside a register marked R/W there could be read-only bits.

Table 158. Register overview: flash (base address = 0x40034000)

Name	Access	Offset	Description	Reset value	Section
CMD	W	0x0	Command register	0x0	5.6.1.1
EVENT	W	0x4	Event register	undefined	5.6.1.2.4
STARTA	RW	0x10	Start (or only) address for next flash command	undefined	5.6.1.2.1
STOPA	RW	0x14	End address for next flash command, if command operates on address ranges	undefined	5.6.1.2.1
DATAW0	RW	0x80	Data register, word 0-7, Memory data, or command parameter, or command result.	0x0	5.6.1.2.3
DATAW1	RW	0x84	Data register, word 0-7, Memory data, or command parameter, or command result.	0x0	5.6.1.2.3
DATAW2	RW	0x88	Data register, word 0-7, Memory data, or command parameter, or command result.	0x0	5.6.1.2.3
DATAW3	RW	0x8C	Data register, word 0-7, Memory data, or command parameter, or command result.	0x0	5.6.1.2.3
INT_CLR_ENABLE	W	0xFD8	Clear interrupt enable bits	undefined	5.6.1.3.1
INT_SET_ENABLE	W	0xFDC	Set interrupt enable bits	undefined	5.6.1.3.2
INT_STATUS	RW	0xFE0	Interrupt status bits	undefined	5.6.1.3.3
INT_ENABLE	RW	0xFE4	Interrupt enable bits	undefined	5.6.1.3.4
INT_CLR_STATUS	W	0xFE8	Clear interrupt status bits	undefined	5.6.1.3.5
INT_SET_STATUS	W	0xFEC	Set interrupt status bits	undefined	5.6.1.3.6
MODULE_ID	R	0xFFC	Controller +Memory module identification	0xC40F0800	5.6.1.3.7

5.6 Register description

This section lists the individual bit fields which make up each single register, and their purpose.

A more detailed description for some bit fields can often be found in Section 5.7 “Functional description” references to the specific section(s) are provided.

When a field is marked as *Reserved*, this means that no function is currently assigned to that field. To ensure compatibility with future enhancements, software should not rely on the value read, and should not modify the bit (i.e: writes should confirm the value just read). When reading is not possible (e.g. write-only register) or not practical, the reset value should be written on reserved fields. Typically reserved fields read as 0 and their write data is discarded, but this may not always be the case.

5.6.1 Controller specific registers

Valid APB transactions to all registers specified in this section, with the exception of the EVENT register, stall if accessed when a sequencer command is pending or running. Access to other registers never stalls.

Remark: A command is pending if the bus transaction that started it has already occurred, but the sequencer waits for the completion of an ongoing read operation before starting

5.6.1.1 Command register

The controller manages execution of *commands*. A *commands* is any action performed by the controller, for example, a mode change, programming, erasing, or calculating a checksum over an address range. See Section 5.6.2 “Command listing (CMD)” for a list of available commands.

A command usually has parameters, such as an address or address range, data to be written, a mode specification. Parameters must be written into corresponding registers before that command is started. Writing parameters has no effect until the command is started.

Command execution is triggered when writing the CMD register.

When a command is executed, it sets appropriate bits in the INT_STATUS registers. Some commands also return additional information in other registers.

Note: Associated prefetches should be disabled before issuing any Flash commands for CPU0.

Table 159. Command register (CMD, offset = 0x0)

Bit	Symbol	Description	Reset value
31:0	CMD	Command register.	0x0

5.6.1.2 Parameter or result registers

The following registers hold command parameters and/or command results. DATAWx registers are always updated as a result of executing a controller command, even if the command description does not report a result to be returned on some or all registers. STARTA and STOPA only contain parameters, and are never updated by a running command.

STARTA and STOPA are used for the command to specify the start and end address. This register contains the address in units of memory words and not bytes.

This is a physical word address inside the flash memory (that is, address 1 represents the second 128-bit word inside the flash memory, not the second byte in the first word

5.6.1.2.1 Start address register

Table 160. Start (or only) address for next flash command (STARTA, offset = 0x10)

Bit	Symbol	Description	Reset value
17:0	STARTA	Address / Start address for commands that take an address (range) as a parameter.	0x0
31:18	-	Reserved. Read value is undefined, only zero should be written.	undefined

5.6.1.2.2 Stop address register

Table 161. End address for next flash command, if command operates on address ranges (STOPA, offset = 0x14)

Bit	Symbol	Description	Reset value
17:0	STOPA	Stop address for commands that take an address range as a parameter (the word specified by STOPA is included in the address range).	0x0
31:18	-	Reserved. Read value is undefined, only zero should be written.	undefined

5.6.1.2.3 Data register

Table 162. Data register, word 0-3, Memory data, or command parameter, or command result. (DATAW0-3, offset = 0x80 to 0x08C

Bit	Symbol	Description	Reset value
31:0	DATAW	Memory data, or command parameter, or command result.	0x0

5.6.1.2.4 Event register

As a general rule, when the controller is busy executing a command it is not possible to give further orders, and all registers involved in command execution cannot be used (an access would stall the APB bus).

However, some events may happen (also) when the controller is busy executing a command, and these events would influence the command being executed. Examples of such events are a reset, a command abort request, wake-up from power-down.

The event register allows to generate such events through software.The event register is write-only. The act of writing the register with one of the bits at 1 activates the generation of the corresponding event.

Table 163. Event register (EVENT, offset = 0x4)

Bit	Symbol	Description	Reset value
0	RST	When bit is set, the controller and flash are reset.	0x0
1	WAKEUP	When bit is set, the controller wakes up from either low power or power-down mode that was active.	0x0
2	ABORT	When bit is set, a running program/erase command is aborted.	0x0
31:3		Reserved. Read value is undefined, only zero should be written.	undefined

5.6.1.3 Interrupt and Identification registers

5.6.1.3.1 Interrupt registers

These interrupt registers determine when the controller gives an interrupt request. The interrupt line output is asserted when the bit-wise AND of INT_STATUS and INT_ENABLE is nonzero.

If the corresponding INT_ENABLE bit is zero, an INT_STATUS register bit can be polled to test for the occurrence of an event.

The INT_STATUS register can be set for software testing purpose, by writing into the INT_SET_STATUS register.

Table 164. Clear interrupt enable bits (INT_CLR_ENABLE, offset = 0xFD8)

Bit	Symbol	Description	Reset value
0	FAIL	When a CLR_ENABLE bit is written to 1, the corresponding INT_ENABLE bit is cleared.	0x0
1	ERR	When a CLR_ENABLE bit is written to 1, the corresponding INT_ENABLE bit is cleared.	0x0
2	DONE	When a CLR_ENABLE bit is written to 1, the corresponding INT_ENABLE bit is cleared.	0x0
3	ECC_ERR	When a CLR_ENABLE bit is written to 1, the corresponding INT_ENABLE bit is cleared.	0x0
31:4		Reserved. Read value is undefined, only zero should be written.	undefined

5.6.1.3.2 Set interrupt enable bits register

Table 165. Set interrupt enable bits (INT_SET_ENABLE, offset = 0xFDC)

Bit	Symbol	Description	Reset value
0	FAIL	When a SET_ENABLE bit is written to 1, the corresponding INT_ENABLE bit is set.	0x0
1	ERR	When a SET_ENABLE bit is written to 1, the corresponding INT_ENABLE bit is set.	0x0
2	DONE	When a SET_ENABLE bit is written to 1, the corresponding INT_ENABLE bit is set.	0x0
3	ECC_ERR	When a SET_ENABLE bit is written to 1, the corresponding INT_ENABLE bit is set.	0x0
31:4		Reserved. Read value is undefined, only zero should be written.	undefined

5.6.1.3.3 Interrupt status bits register

Table 166. Interrupt status bits (INT_STATUS, offset = 0xFE0)

Bit	Symbol	Description	Reset value
0	FAIL	This status bit is set if execution of a (legal) command failed.	0x0
1	ERR	This status bit is set if execution of an illegal command is detected.	0x0
2	DONE	This status bit is set at the end of command execution.	0x0
3	ECC_ERR	This status bit is set if, during a memory read operation (either a user-requested read, or a speculative read, or reads performed by a controller command), the ECC decoding logic detects a correctable or uncorrectable error.	0x0
31:4		Reserved. Read value is undefined, only zero should be written.	undefined

5.6.1.3.4 Interrupt enable bits

Table 167. Interrupt enable bits (INT_ENABLE, offset = 0xFE4)

Bit	Symbol	Description	Reset value
0	FAIL	If an INT_ENABLE bit is set, an interrupt request will be generated if the corresponding INT_STATUS bit is high.	0x0
1	ERR	If an INT_ENABLE bit is set, an interrupt request will be generated if the corresponding INT_STATUS bit is high.	0x0
2	DONE	If an INT_ENABLE bit is set, an interrupt request will be generated if the corresponding INT_STATUS bit is high.	0x0
3	ECC_ERR	If an INT_ENABLE bit is set, an interrupt request will be generated if the corresponding INT_STATUS bit is high.	0x0
31:4		Reserved. Read value is undefined, only zero should be written.	undefined

5.6.1.3.5 Clear interrupt status bits

Table 168. Clear interrupt status bits (INT_CLR_STATUS, offset = 0xFE8)

Bit	Symbol	Description	Reset value
0	FAIL	When a CLR_STATUS bit is written to 1, the corresponding INT_STATUS bit is cleared.	0x0
1	ERR	When a CLR_STATUS bit is written to 1, the corresponding INT_STATUS bit is cleared.	0x0
2	DONE	When a CLR_STATUS bit is written to 1, the corresponding INT_STATUS bit is cleared.	0x0
3	ECC_ERR	When a CLR_STATUS bit is written to 1, the corresponding INT_STATUS bit is cleared.	0x0
31:4		Reserved. Read value is undefined, only zero should be written.	undefined

5.6.1.3.6 Set interrupt status bits

Table 169. Set interrupt status bits (INT_SET_STATUS, offset = 0xFEC)

Bit	Symbol	Description	Reset value
0	FAIL	When a SET_STATUS bit is written to 1, the corresponding INT_STATUS bit is set.	0x0
1	ERR	When a SET_STATUS bit is written to 1, the corresponding INT_STATUS bit is set.	0x0
2	DONE	When a SET_STATUS bit is written to 1, the corresponding INT_STATUS bit is set.	0x0
3	ECC_ERR	When a SET_STATUS bit is written to 1, the corresponding INT_STATUS bit is set.	0x0
31:4		Reserved. Read value is undefined, only zero should be written.	undefined

5.6.1.3.7 Identification register

The purpose of this read-only register is to give information over the controller version

Table 170. Controller and Memory module identification (MODULE_ID, offset = 0xFFC)

Bit	Symbol	Description	Reset value
7:0	APERTURE	Aperture i.	0x0
11:8	MINOR_REV	Minor revision i.	0x8
15:12	MAJOR_REV	Major revision i.	0x0
31:16	ID	Identifier.	0xC40F

5.6.2 Command listing (CMD)

This section lists all commands that can be specified in the CMD register. Irrespective of how command execution is triggered, any ongoing memory read is completed before that actual command execution starts. When command execution is triggered but not yet started, the command is said to be pending.

When any command completes execution, it sets the DONE bit in the INT_STATUS register. All commands report failure and error status bits as specified in their respective description; such flags are not listed in the command's output results. In general, when an error is detected (either by command execution, or because no command could be executed) no command result is defined, not even a fail status. Therefore, if a command (or a CMD register write operation) sets the INT_STATUS ERR bit, it will not modify the FAIL bits and the result registers.

Remark: All registers capable of holding a command result (DATAWx) are always updated by a running command. If no specific result is listed for any of these registers, its content after command execution is undefined.

When a register (STARTA, STOPA, DATAWx…) is said to contain an address, this is a physical word address inside the flash memory (that is, address 1 represents the second 128-bit word inside the flash memory, not the second byte in the first word). When a page address is required/returned, the 5 least significant bits of the address are don't-care (a flash page contains 32 user-accessible words).

Addresses and address ranges given as parameters have to be within the address range of the memory.

Table 171. CMD listing

Command	Value	Parameters	Output	Description
CMD_INIT	0	None	None	Initialization. Automatically triggered when exiting from Reset.
CMD_POWERDOWN	1	DATAW0 (See Table 172)		When this command is started, the flash and controller enter power-down mode. During power-down (as with any other command), the flash is not accessible. The power-down command waits indefinitely for a wake-up event. When such an event happens (triggered by the EVENT register), the controller will disable flash power-down and then will wait until the flash is ready for operation, with a time-out of 4096 clock cycles; FAIL is reported if the time-out is reached. Then the command terminates
CMD_SET_READ_MODE	2	DATAW0 (see Table 172)	None	The flash data sheet reports the minimum duration of the pre-charge (Tp) and evaluation (Tdpd) as a function of the memory size, and of whether EWLE is active or not. Select the figures for EWLE=1, sum them up, add ~34ns (to take address path and ECC delay, wire delay, jitter, read delay uncertainty, data setup... into account: exact value is to be determined after synthesis), then divide the result by the clock period, rounding down the division result to an integer: this will give the values to specify in bits 3:0 of the DATAW0 register. The clock frequency should be kept constant while a controller command is being executed.
CMD_READ_SINGLE_WORD	3	STARTA (flash word address), DATAW0 (read mode). (See Table 173)	DATAW0-3: Read Data	This command reads a single memory word, using a specified combination of read modes. For instance, it is possible to perform a read of the DMACC word with ECC disabled.The controller will respond to the command by setting the ERR flag if an illegal mode combination is requested. Depending on the chosen modes, the controller ensures that adequate settling times are met, both when the modes are activated and when they are deactivated.

Table 171. CMD listing *…continued*

Command	Value	Parameters	Output	Description
CMD_ERASE_RANGE	4	STARTA, STOPA	None	The range from the page containing the STARTA address to the page containing STOPA (included) is erased. An abort event interrupts erasing, unless the event happens very late in the erase process (when the flash is discharging high voltages and reconfiguring itself for reading), where it would have no effect. If abort influences the erase process, the FAIL flag is set. When erasing completes, the controller waits until the flash is ready for operation, with a time-out of 4096 clock cycles; FAIL is reported if the time-out is reached. Then the command completes. The FAIL flag is also set in the case the flash reports an HV error (requested high voltages could not be reached). If STARTA points to a page following the one pointed by STOPA, no page is erased and the ERR flag is set.
CMD_BLANK_CHECK	5	STARTA, STOPA	DATAW0 contains address inside the first failing page (if any). If the FAIL flag is not set, the content of DATAW0 is not significant. Do not assume that this is the address of the first failing word; in the case of a DMACC word failure, such an address would not be representable.	The range from the page containing address STARTA to the page containing STOPA (included) is checked. The selected pages are checked for the erased condition (all0 including parity), with a specific margin read mode. ECC is off during the check (single bit errors cause failures). If a page is found which is not correctly erased, the FAIL flag is set, the page is reported on DATAW0 and processing stops. The check is performed in incrementing address order, so that, in case of fail, it is known that pages at a lower address than the failing one are successfully verified. To know the individual status of all selected pages, when a fail is reported on a page which is not the last in the range, the command should be restarted with the page following the failing one being selected as start page. Checking a page range is more time-efficient than individually running the check command on single pages. If STARTA points to a page following the one pointed by STOPA, no page is checked and the ERR flag is set. As a side effect of this command, the ECC log is cleared. This is because the same HW resources are used to record the failing page.
CMD_MARGIN_CHECK	6	STARTA, STOPA	DATAW0: an address inside the first failing page (if any).	This command checks the selected page range for correct programming. If, for any reason, programming was interrupted or disturbed, or erase was performed without a subsequent programming, this check fails.
CMD_CHECKSUM	7	STARTA, STOPA	DATAW0-3, the computed checksum	

Table 171. CMD listing *…continued*

Command	Value	Parameters	Output	Description
CMD_WRITE	8	STARTA, DATAW0-3: word to be written	None	The selected word is copied into the page register, at the specified position. STARTA is the column address of the word to be written.
CMD_WRITE_PROG	10	STARTA, DATAW0-3: word to be written	None	This command first performs a “write word” command, then, if the written word was the last of a page, it performs a “program page” command.
CMD_PROGRAM	12	STARTA	None	First, an all1 value (data+parity) is stored in the page register in the location corresponding to the DMACC word(*). Then, programming is started, which copies the page register content into the selected page. The controller waits until the flash is ready for operation, with a time out of 4096 clock cycles; FAIL is reported if the time out is reached.
CMD_REPORT_ECC	13	None	DATAW0: address of first word with ECC event DATAW1: number of uncorrectable errors found DATAW2: number of corrections performed	All ECC events are logged, both for reads performed by user code and for internally-generated reads (e.g. checksum and “read word” commands, initialization…). This command copies logging information to the DATAW0-2 registers, and then clears the log, zeroing the counters. 20-bit counters are used. When they reach their maximum value, further incrementing is prevented (that is, they saturate rather than wrapping around). As the DMACC word is not meant to contain ECC-encoded data, ECC errors are not logged for it.

Table 172.

Bit	Function
[31-4]	Reserved. Do not modify.
[3-0]	Number of extra wait states for controller-internal reads.

Table 173.

Bit	Function
[15]	Read DMACC word.
[14-12]	Reserved.
[11-10]	00: normal read. 01: margin vs program. 10: margin vs erase. 11: illegal bit combination.
[9-3]	Reserved.
[2]	Read with ECC off.
[1-0]	Reserved.

5.7 Functional description

This chapter contains the following information:

Throughout the chapter, [pseudo] code examples are also given. In these examples, it is assumed that register names are accessible through variables with the same name. Syntax is pseudo-C language.

- Detailed specification of the behavior of the controller (with the exception of commands, which are described in Section 5.6.2 “Command listing (CMD)”.
- Constraints that must be followed while using the controller If these constraints are not met, the controller and/or the associated memory will not behave as specified.
- Instructions for the use of the controller (including usage examples), explanation of the rationale behind the architectural choices, caveats and warnings.

5.7.1 Basic principles of operation

This section lists information which is common to multiple controller functions.

5.7.1.1 Definitions

The memory managed by the controller can execute the following basic operations:

- Reading: it is the process of extracting the information contained at a specific memory location
- Writing: it is the process of updating temporary storage present in the memory, called *page register*, with data that must subsequently be programmed
- Program/erase: it is the process by which the memory will alter its nonvolatile content, by either clearing all selected bits to a default value (erase), or setting them to the value specified by the page register (program).
- Power down: the memory is put in a mode where a minimum amount of supply current is used; no (other) operation can be performed in this state
- When none of these operations is being performed, the memory is said to be idle.

In general, read and write operations on the memory are de-coupled by read and write requests to the controller: a single controller command can perform multiple memory operations.

In any case, no flash operation is initiated without a triggering event (e.g. a write to the CMD register, activation of the memory read, or a reset).

5.7.2 Address validity

If a nonexisting memory location is addressed through the flash read address, the read result is unspecified.

If a read or write operation is performed on a nonexisting register address, writes are ignored and reads return 0; it is unspecified whether such access would stall.

Remark: If an access to an existing register address would have stalled, then it is possible that access to a nonexisting register address stalls as well.

If a read operation is performed on a write-only register (for example, without the “R” specifier in the Access column of Table 158), undefined data is returned.

If a write operation is performed on a read-only register (for example, without any of the “W”, “S”, orc “C” specifier in the Access column of Table 158, the operation is ignored.

If an address is used with bits 1-0 not 00 on APB, a bus error is reported.

5.7.3 Initialization

When entering the reset mode (hard reset, or “1” written into the RST bit of the EVENT register), all controller registers will be initialized to the value specified in the relative register description. Any command or bus transaction in progress is interrupted as well, with no regards for data integrity.

Immediately after leaving reset mode, an initialization phase takes place, where some memory locations are read, and corresponding volatile locations are initialized depending on the value just read.

The controller reads 19 locations in the last two pages of the flash (see Table 158 for the exact locations and their content). For each location read, it initializes the corresponding volatile storage in the controller: flash trim values, flash repair info, gpo trim bus.

If an un-correctable ECC error is detected, the corresponding volatile storage is not updated (so that the safe default values are kept), and the initialization is immediately terminated with the FAIL flag set.

A per-page checksum protects the integrity of information read by the Initialization command. An additional word is programmed with a value, such that the checksum of the words read (including the additional word) is 0. The checksum algorithm is the same as the one used by the *checksum page range* command.

If initialization reports a FAIL, the flash was not correctly configured, and must not be used (read data may be incorrect, and writing may corrupt the content). Security-conscious users should ensure that an application is not started with a failing init.

Although the controller will not ensure that reading is performed correctly and with the correct mode in case of an init error, reading is anyway permitted, to avoid ending up with inaccessible samples in the case of initialization issues.

5.7.4 Configuration

Controller configuration amounts to specifying options such as read speed, and caching/pre-fetching options in a way that best matches system operation. Configuration is normally performed by system software shortly after initialization, although default configuration values are normally chosen to allow safe operation with no further software intervention. When conditions change (for example, the system clock frequency is changed), configuration can be repeated.

Be aware that configuration errors may prevent correct working of the flash.

This is an example of code to perform configuration just after exiting from reset. It assumes execution from ROM by default, with comments specifying the differences in the case of booting from flash.

```
//check init status. Not needed if booting from flash: in that case, the safest
//option is to prevent fetching from flash if pin init_error=1.
while(!(INT_STATUS & 0x4 )) ; //wait until DONE is set. Not needed if CPU reset
//is only released when ctl_busy=0
if(INT_STATUS & 0x1) {
handle_boot_error(); //communicate to the external world that a
//non-recoverable error occurred.
while(1); //handle_boot_error should not return. In any case, the flash
//cannot be used.
}
//end of init status checking
//begin interrupt configuration
INT_CLR_ENABLE = 0x1f;//clear all interrupt enables. Not needed just after
//reset, as this is the default state
INT_SET_ENABLE = 0x2; //only enable interrupt on ERR status.
//Correct code would never set this flag.
//Correctable ECC events can be managed by periodic checks.
//Most examples in this manual will poll the DONE bit
//and explicitly check for FAIL status, so no INT on these.
//end interrupt configuration
//begin of read mode configuration
//EXAMPLE CODE! values may also depend on target clock frequency
fmc_cache_controller_config = flash_location_containing_cache_controller_default_WS;
DATAW0 = flash_location_containing_flash_controller_default_WS;
CMD = CMD_SET_READ_MODE; //this starts the "set read modes" command
//no need to wait until command is completed: further accesses are stalled
//until the command is completed.
switch_to_target_clock_frequency();
//end of read mode configuration
//begin of program/erase configuration [optional, see 8.7.3]
DATAW0 = 0xf; //slowest clock for both program and erase
CMD = CMD_SET_WRITE_MODE; //no need to wait for completion
//end of program mode configuration
```

5.7.5 Memory power-down

In this controller, power-down is implemented as a command. See Section 5.6.2 "Command listing (CMD)" for details on the power-down command.

During power-down, the memory will be placed in a mode where it draws a minimum amount of current.

During power-down (as with any other command) non-volatile memory controller is busy performing a command., and all memory read requests will be ignored.

Power-down is exited by a wake-up event, which can be triggered by writing a 1 in the WAKEUP bit of the EVENT register.

Power-down is also exited in case of a reset.

After that a wake-up event is triggered, the controller will wait for the memory to recover, and then end the power-down command, thus re-enabling read.

5.7.6 Codes examples

In this example, powerdown is used as a low-power version of the CPU's WFI instruction. For this example to work, code is executed from flash, and the interrupt controller activates the *wake-up* input of a flash controller if a valid interrupt request comes.

```
Enable_interrupt_sources(); //to be sure that wake-up will occur
CMD = CMD_POWERDOWN;
//Now the CPU will try to fetch the next word from the flash, which will stall
//because the flash is in powerdown mode. Whenever an interrupt request comes,
//the pending read will be completed, then (if CPU interrupts are enabled) the
//interrupt service routine is executed, then the following code is executed:
Process_interrupt_event();
```

In the following example, the CPU determines that it does need the flash for a while (all needed code is in ROM/RAM), then turns it off temporarily. Be sure not to access flash when it is in power-down mode, otherwise the system will hang (a watchdog timer is recommended).

```
//executing from ROM/RAM:
INT_CLR_STATUS = 0x4; //clear the DONE status bit
CMD = CMD_POWERDOWN;
do_things_without_flash();
//when we need the flash again:
EVENT = 0x2; //WAKEUP event
while(!(INT_STATUS & 0x4 )) ; //wait until DONE is set
do_things_with_the_flash();
```

In the above example, INT_STATUS register handling can be removed, if it is OK that, in case the flash is accessed after wakeup is triggered but before the flash is ready, the system may temporarily be stalled.

5.7.7 Reading

The memory is read through the AHB bus. Normal user memory is mapped on the AHB address space, as a contiguous address space, starting from address 0.

The Flash contains one additional word per page (the so-called “dmacc” word). Such words are not readable through the AHB bus. These words are managed internally by the controller in order to store a flag (all1), which can be used to verify whether a programming operation was prematurely terminated. See Section 5.6.2 “Command listing (CMD)”

Reading is not possible if the controller is executing a command.

5.7.8 Writing

For writing, a number of APB writes are needed to fully define a memory word, which is larger than 32 bits. The controller accumulates data inside its own internal storage, until the content of a full memory word has been specified. When this is done, the full word is transferred to the memory's page register (at the position specified by the STARTA register), as a single operation.

Data to be written is accumulated inside the controller's DATAW0-DATAW3 registers.

After specifying an address in the STARTA register and 128 bit of data in the DATAW0-3 registers, it's possible to activate the controller's *Write"* command, which will transfer the data to the memory's page register, at the position indicated by the STARTA register (only the column part of the address is significant).

5.7.9 Erasing, programming, and verifying

Some controller commands can modify the content of the memory: program page, erase and page range. Other commands are targeted at verifying the content of the memory: checksum address range, blank check and margin check. Such commands operate either on a single address, specified by the STARTA register, or on an address range, specified by both STARTA and STOPA. Since all memory program/erase operations have a page granularity, column address bits are don't-care in the case of program, erase, and some other commands.

Additional command parameters may be required (see the command documentation for details): they can be written in the DATAWx registers. Writes in STARTA, STOPA, and DATAWx registers can happen in any order, and have no other effect than modifying the register's content.

When all command parameters are set, the command can be started, by writing the command's code into the CMD register.

During command execution, controller is busy, and access to some registers (CMD, STARTA, STOPA, DATAWx) is stalled. Other registers remain accessible: it is therefore possible to poll the INT_STATUS register and change INT_ENABLE; it is also possible to force an ERR or FAIL indication through writing to INT_SET_STATUS, in order to test the application's behaviour in the case of an error condition.

5.7.10 Code examples

This section presents an example of pseudocode to copy two pages (1024 bytes) of code from address src to address dst. Address dst is relative to the beginning of the flash address space, and is page-aligned (that is, a multiple of 512).

This code demonstrates the erase, write and program commands. If this code is not fetched from the flash itself (that is, it is fetched from RAM/ROM), accesses to the flash and controller never stall, therefore other masters are not prevented from accessing other resources on the bus. Interrupts are not needed, but they can be enabled and, as long as their service routines do not try to access the flash and controller, they retain their real-time performance.

```
int *src_i;
INT_CLR_STATUS = 0xf; //clear status register
STARTA = ((int)dst)>>4; //set start address. Assuming dst is a char*
STOPA = STARTA+32; //set end address. 1 page = 32 flash words.
CMD=CMD_ERASE_RANGE; //command: erase page range. Now erase starts.
while(!(INT_STATUS & 0x4 )) ; //wait until DONE is set
if(INT_STATUS & 3) handle_erase_errors();
//now write & program
src_i = (int *) src;
for(page=0; page < 2; page++) {
for(flashword=0; flashword<32; flashword++) {
```

```
INT_CLR_STATUS = 0xf; //clear status register
STARTA = flashword;
DATAW0 = *src_i++;
DATAW1 = *src_i++;
DATAW2 = *src_i++;
DATAW3 = *src_i++;
CMD=CMD_WRITE; //start write
while(!(INT_STATUS & 0x4 )) ; //wait until DONE is set
if(INT_STATUS & 3) handle_write_errors();
} //end of word loop
INT_CLR_STATUS = 0xf; //clear status register
STARTA = (((int)dst)>>4) + page*32
CMD=CMD_PROGRAM; //start program
while(!(INT_STATUS & 0x4 )) ; //wait until DONE is set
if(INT_STATUS & 3) handle_program_errors();
} //end of page loop
```

5.7.11 Command abort

Some commands can be aborted while they are executing; this is normally the case for erase and program commands, which take a long time: the application may need to urgently access the memory, and then it needs to abort the command being executed.

An abort event can be specified through the ABORT bit of the EVENT register.

An aborted command flags unsuccessful completion by setting the FAIL bit in the INT_STATUS register. A failed program/erase command has to be retried, even if the memory content appears to be OK (either the original one or the new one).

An abort request during the execution of a command that can be aborted does not necessarily result in a FAIL indication: when the request arrives very late in the command execution timeframe (i.e. when the command is already busy restoring safe read conditions) the request is ignored.

5.7.12 Verification

The flash and controller offer a number of commands to check whether the memory has been correctly programmed or erased. As a rule, there is no need to run any type of verification after programming or erasing, except for safety applications where it is desired that the consequence of an error is known/deterministic. However, such commands come handy in order to verify whether a flash content modification has been allowed to complete successfully (for instance, a reset or power loss could interrupt an ongoing operation). In the following, a simple example of how a small amount of data which is modified often can be handled in order to guarantee that, in case of power loss during a modification, valid data can be always retrieved (either the old data or the new data).

In this example, the size of the data to be stored fits in a single flash page, leaving some room for locations required for algorithm management. Two pages are used: one normally contains the data, while the other is erased. When writing, new data is firstly programmed in the erased page, then old data is erased. The get_data() function returns the address of the page which contains valid data, performing cleanup of the other page if necessary

(cleanup is necessary if programming or erasing was interrupted: in this case, one of the pages contains valid data while the other holds data halfway between programmed and erased levels). The put_data() function updates the stored data.

The concepts shown in this example can be adapted to different contexts (e.g. different data sizes), and optimizations can be performed (e.g. caching get_data() [intermediate] results to RAM, using multiple blank pages with one data page [to increase cycling endurance], sharing one backup page with multiple data pages [to reduce flash space – do not use a fixed backup page, otherwise it will be cycled too quickly], etc…).

```
const char *page0 = address_of_1st_flash_page;
const char *page1 = address_of_2nd_flash_page;
char *get_data()
{
//DMACC words are all_0 for an erased page, all_1 for a programmed page
//doing a quick sanity check, to avoid time-consuming checks if not needed
//get_dmacc_status() returns 0 for all0, 1 for all1, 2 for any other content
int page0_status=get_dmacc_status(page0);
int page1_status=get_dmacc_status(page1);
if(page0_status==1 && page1_status==0) return page0;
if(page0_status==0 && page1_status==1) return page1;
//if we are here, the status of pages is not ideal... check full pages
//get_page_status returns 0 for a fully erased page, 1 for a correctly
//programmed page, 2 for a corrupted page
page0_status=get_page_status(page0, page0_status);
page1_status=get_page_status(page1, page1_status);
if(page0_status==2 && page1_status==2)
return do_recover(); //both pages marginal or KO
//at least one page is good (it's not possible that both are erased)
if(page0_status==1) {erase(page1); return page0;}
else {erase(page0); return page1;}
}
int get_dmacc_status(char *page)
{
int res;
STARTA = ((int)page)>>4;
DATAW0 = 0x8004; // read DMACC word, normal mode, ECC off
CMD= CMD_READ_SINGLE_WORD;
//the following access to DATAW0 is automatically stalled until the command completes
res=(DATAW0==0xffffffff)?1:(DATAW0==0)?0:2;
if(DATAW1!=DATAW0) return 2;
if(DATAW2!=DATAW0) return 2;
if(DATAW3!=DATAW0) return 2;
return res;
}

int get_page_status(char *page, int hint)
{
int res;
if(hint==2) return 2; //margin checks surely fail if usermode read is wrong
INT_CLR_STATUS=0x7; //clear DONE FAIL ERR (ERR is optional)
STARTA = STOPA = ((int)page)>>4;
```

```
CMD= hint? CMD_MARGIN_CHECK: CMD_BLANK_CHECK; //run the right command
while(!(INT_STATUS & 0x4 )) ; //wait until DONE: needed as INT_STATUS doesn't stall
return (INT_STATUS & 1)? 2: hint; //if the command does not fail, the hint was correct
}

void erase(char *page)
{
INT_CLR_STATUS=0x7; //clear DONE FAIL ERR (ERR is optional)
STARTA = STOPA = ((int)page)>>4;
CMD= CMD_ERASE_RANGE;
while(!(INT_STATUS & 0x4 )) ; //wait until DONE: needed as INT_STATUS doesn't stall
if(INT_STATUS&1) handle_hw_failure(); //erase of 1 page is always meant to pass
}

char *do_recover()
{
// check with the help of user_mode & signatures whether one of the pages still has
//valid data; re-write it to get better margin
char *good_page=NULL;
char *other_page;
char buf[512];
if(consistency_check(page0,buf)) good_page=page0;
else if (consistency_check(page1,buf)) good_page=page1;
if(!good_page) {
handle_hw_failure(); //we don't get consistent data anywhere
return NULL;//best would be that handle_hw_failure() does not return at all
}
//don't overwrite the ~good page, use the other one
erase(other_page);
program(buf,other_page); //the data which was previously read and found consistent
//is used to re-program; if we re-read good_page (which
//failed margin checks), we might get different data!
erase(good_page);
return other_page;
}
int consistency_check(char *page,char *buf)
{
int i;
int *ip; //assuming 32-bit integer
//If the optional check on many ECC corrections is performed (see below),
//this fragment of code is best executed from RAM/ROM, with other bus masters
//disabled, in order to avoid that other accesses cause additional ECC corrections
CMD=CMD_REPORT_ECC; //clear ECC datalog
ip=(int *)buf;
for(i=0;i<32;i++) { //32 words in a page
//use READ_SINGLE_WORD command to avoid bus errors on corrupt data
STARTA = ((int)page)>>4;
DATAW0 = 4; //read with ECC off
CMD= CMD_READ_SINGLE_WORD;
*ip++=DATAW0;
*ip++=DATAW1;
```

```
*ip++=DATAW2;
*ip++=DATAW3;
page+=16; //a word contains 16 bytes
}
CMD=CMD_REPORT_ECC; //get ECC datalog
//end of execution from RAM/ROM
if(DATAW1) return 0; //fail if there are uncorrectable words
//optional, if it's more risky to process dubious data than to report a data loss:
//if(DATAW2>treshold) return 0; //avoid too many corrections as well
return check_user_consistency(buf); //check based on the structure of the user payload

//For example, a checksum may have been added to the data; some values might be known
//to be within specific ranges; some fixed-content fields may be there; etc...
//Note that the check is performed on the buffer, not directly on the flash.
}

void put_data(char *src)
{
char *old_page=get_data();
char *new_page=(old_page==page1)? page0:page1;
//new_page is the page NOT returned by get_data(), the other one (expected blank)
INT_CLR_STATUS=0x7; //clear DONE FAIL ERR (ERR is optional)
STARTA = STOPA = ((int)new_page) >>4;
CMD= CMD_BLANK_CHECK; //needed to ensure that erase was properly completed: only
// the DMACC word was possibly checked
while(!(INT_STATUS & 0x4 )) ; //wait until DONE: needed as INT_STATUS doesn't stall
if(INT_STATUS&1) erase(new_page);
program(src,new_page); //copy the code into the new page. For examples on how to
                       //do this, see 8.7.1 (program code only)
erase(old_page);
}
```

The following example is targeted at verifying the correctness/integrity of a code area; it can be used for example after an application upgrade, or periodically to ensure that the correct code is still there (for example, not modified by a hacker, a programming error, and an HW failure). The area is delimited by start_address and end_address (end_address still included in the range). The content programmed in that page range has a known 128-bit checksum. Other than verifying the checksum, this example checks whether a high number of ECC corrections were found (an unexpected ECC uncorrectable error results in a failing checksum; *expected* errors can occur if erased pages are included in the checked range).

```
//execute this code from RAM/ROM, so that fetching does not create
//additional ECC errors
CMD= CMD_REPORT_ECC; //clear ECC error count
STOPA = ((int)end_address) >> 4;
STARTA = ((int)start_address) >> 4;
CMD=CMD_CHECKSUM;
//the following access will stall until the checksum command is completed.
//if this is not desired, then either poll the DONE bit in the INT_STATUS register,
//as in previous code examples, or configure an interrupt to occur on DONE
//and wait for it before proceeding with execution.
```

```
if(DATAW0!=known_checksum[0] || DATAW1!=known_checksum[1] || DATAW2!=known_checksum[2]
|| DATAW3!=known_checksum[3]) return CHECK_FAILED;
CMD= CMD_REPORT_ECC; //get ECC error count
if(DATAW2>ECC_correction_treshold) return CHECK_FAILED; //singlebit corrections
return CHECK_PASSED;
```

5.7.13 ECC

The ECC function is normally transparent to the user.

When writing, parity is automatically computed and stored alongside user data.

When reading, data and parity are used to reconstruct correct data, even in the case of a 1-bit error.

ECC has to be taken into account only in the following contexts:

- In case of a correction or uncorrectable error, this fact and the location of the error are logged inside the controller, and an interrupt is optionally generated: in case of failure the application must be able to take countermeasures, and even if execution is not endangered (when a correction is successfully performed), the application may choose to refresh memory data to avoid that a subsequent error in the same word causes a failure.
- Flags are made available, alongside read data and with the same timing, to identify ECC corrections and uncorrectable errors.
- When reading an erased location, an uncorrectable error is flagged. Use the “blank check” command to test for successful erase.
- Due to the presence of ECC, over-programming an already programmed word will likely result in inconsistent parity bits; for this reason, it is not allowed to program a memory word without erasing it first.
- If a program or erase operation is aborted, data and parity bits are unknown and probably inconsistent: the resulting ECC operation may result in not easily understandable behaviour (for example, when partially erasing a word, a bit which was previously already erased may be read as programmed, due to an inconsistent value of the parity bits).

Each data word has its associated parity bits, and only one wrong bit in the whole word (either in the data or in the parity) can be corrected. When more than one bit is wrong, the read result is unspecified (it is possible that no error is flagged, or that a correctable error is flagged, or an uncorrectable error is flagged).

Whenever a memory word is read by the controller, and a (correctable or uncorrectable) ECC error is identified, the address of the first occurrence of the most severe type of error is captured inside the controller; all errors (correctable or uncorrectable) are separately counted (a saturating counter is used). A controller command allows to read this information and contextually clear the logging information.

ECC is stored inverted, so that an ALL0 or ALL1 output from the memory is flagged as an uncorrectable error. This helps for safety and security, since most (hacker-induced) failures have a common-mode effect on all output bits.

5.7.14 Interrupts

There is a status register bit for each interrupt source, which is automatically set when the corresponding event occurs.

Each interrupt status bit has a corresponding interrupt enable bit; if the interrupt enable and status bits for at least one interrupt source are both set, an interrupt will be raised to the CPUs (as long as the interrupt line number 0 is enabled inside the CPU registers).

The interrupt enable and status register bits are not writable directly: they are set by writing a 1 in the corresponding bit of the INT_SET_ENABLE and INT_SET_STATUS registers respectively, and are cleared by writing a 1 in the corresponding bit of the INT_CLR_ENABLE and INT_CLR_STATUS registers respectively.

If an enabled interrupt event occurs while the corresponding status register bit is being cleared, the interrupt request to the CPUs is set high for at least one clock cycle.

The above provision is to ensure that no event is lost, in case a new event occurs just before the CPU writes the INT_CLR_STATUS register. However, in this case an interrupt can be triggered but it is not possible to determine its source among the ones available within the controller, since the status register would be cleared. It can be normally assumed that this is an ECC interrupt, since software is expected to first clear the indication of the completion status of a command, and only afterwards start a new operation of the same kind. The presence of an ECC error can be confirmed by clearing the ECC log information maintained inside the controller: in case that an ECC error indication is cleared in the status register before being processed, its presence would still be recorded in the ECC log info.

For system reasons, the interrupt request to the CPUs must be kept active until an interrupt service routine handles the interrupt, then the status register must be cleared while interrupts are disabled (either by means of the corresponding INT_ENABLE bit, or through some other viable means).

6.1 How to read this chapter

This chapter applies to all LPC55S6x/LPC55S2x/LPC552x parts.

6.2 Features

128 kB on-chip boot ROM with bootloader that allows various boot options and APIs:

- Based on ISP pins or CMPA setting in PFR region (see: Chapter 10 “LPC55S6x/LPC55S2x/LPC552x Protected Flash Region”), supports automated booting from internal flash.
- IAP calls. See Chapter 8 “LPC55S6x/LPC55S2x/LPC552x ISP and IAP”.
- FLASH API for programming internal flash. See Chapter 5 “LPC55S6x/LPC55S2x/LPC552x Flash””.
- Supports SPI flash recovery boot from 1-bit SPI flash device (revision 1B only). See Section 6.4.3 for more details.

6.3 General description

The internal ROM memory is used to store the boot code. After a reset, the Arm processor starts its code execution from this memory. The bootloader code is executed every time the part is powered-ON, is reset, or wakes up from a deep power-down while in a low power mode.

Images must be stored in internal flash because the LPC55S6x/LPC55S2x/LPC552x has internal flash for code and data storage. The code is then validated, and the boot ROM vectors to on-chip flash.

Depending on the values of the CMPA bits, ISP pin, and the image header type definition, the bootloader decides whether to boot from internal flash or run into ISP mode. See Section 6.5 “PFR region definitions”. The LPC55S6x/LPC55S2x/LPC552x will read status of the ISP pins to determine boot source. See Table 174.

Table 174. Boot mode and ISP download modes based on ISP pins [1]

Boot mode	ISP0 (PIO0_5 pin)	Description
Passive boot	HIGH	The LPC55S6x/LPC55S2x/LPC552x will look for valid image in the internal flash, if no valid image is found, the LPC55S6x/LPC55S2x/LPC552x will enter ISP boot mode based on DEFAULT_ISP_MODE bits defined in Table 175.
ISP boot	LOW	One of the serial interfaces (UART0, I^{2}C1, SPI3, HS_SPI, USB0, USB1) is used to download image from host into internal flash. The first valid probe message on USART, I^{2}C, SPI or USB locks in that interface.

Table 175. ISP download mode based on DEFAULT_ISP_MODE bits (6:4, word 0 in CMPA)

ISP Boot mode	ISP_MODE_2	ISP_MODE_1	ISP_MODE_0	Description
Auto ISP	0	0	0	The LPC55S6x/LPC55S2x/LPC552x probes the active peripheral from one of below serial interfaces, and download image from the probed peripherals: UART0, I2C1, SPI3, HS_SPI, USB0 or USB1.
USB HID ISP	0	0	1	The USB HID class is used to download an image of the USB0/1 port. (HS-USB is the default USB port). **Remark:** For USB HS ISP mode, 16 MHz crystal is required, and for the USB FS ISP mode, ROM uses FRO.
UART ISP	0	1	0	The UART is used to download the image.
SPI Slave ISP	0	1	1	The SPI slave is used to download the image.
I2C Slave ISP	1	0	0	The I2C slave is used to download the image.
Disable ISP	1	1	1	Disable ISP mode.

[1] To enable USB0-FS ISP mode, see the BOOT_CFG (USB_SPEED. attached PFR excel sheet) needs to be set. USB0 FS ISP mode can only be used when using single hub or when connected directly to PC.

Table 176 shows the ISP pin assignments and is the default pin assignment used by the ROM code that cannot be changed.

Table 176. ISP pin assignments

ISP pin	Port pin assignment
ISP0	PIO0_5
USART ISP mode	
FC0_TXD	PIO0_30
FC0_RXD	PIO0_29
I2C ISP mode	
FC1_SDA	PIO0_13
FC1_SCL	PIO0_14
SPI ISP mode[1]	
FC3_SCK	PIO0_6
FC3_SSEL0	PIO0_4
FC3_MISO	PIO0_2
FC3_MOSI	PIO0_3
HS_SPI_SCK	PIO1_2
HS_SPI_SSEL1	PIO1_1
HS_SPI_MISO	PIO1_3
HS_SPI_MOSI	PIO0_26
SPI Flash Recovery mode[2]	
FC3_TXD_SCL_MISO_WS	PIO0_2
FC3_RXD_SDA_MOSI_DATA	PIO0_3
FC3_CTS_SDA_SSELN0	PIO0_4
FC3_SCK	PIO0_6
USB0 ISP mode	
USB0_VBUS	PIO0_22

Table 176. ISP pin assignments *…continued*

ISP pin	Port pin assignment
USB0_DP	
USB0_DM	
USB1 ISP Mode	
USB1_VBUS	Dedicated pin per package
USB1_DP	
USB1_DP	

[1] SPI ISP mode on Flexcomm 3 is supported on device revision A0 only.

[2] SPI Flash Recovery mode applies to device revision 1B only.

[3] To enable USB0-FS ISP mode, see the BOOT_CFG (USB_SPEED. attached PFR excel sheet) needs to be set. USB0 FS ISP mode can only be used when using single hub or when connected directly to PC.

Figure 11 shows the top-level boot process. The boot starts after Reset is released.

The CPU clock is 48 MHz based on the 96 MHz FRO. When the Cortex-M33 starts the bootloader, the SWD access is disabled and therefore, the debugger is unable to connect to the CPU during this period of time. The boot ROM determines the boot mode based on the reset state of the ISP pins.

After the boot mode is determined, and the image is present in internal flash, the bootloader will validate the vector table and image header.

The boot ROM checks the following for image validity check:

- Validate image using header and CMPA settings when secure boot is enabled. See the Secure Boot chapter for more details.
- Validate image using CRC32 when secure boot is NOT enabled, and the CRC check is enabled in image header.
- Validate the SP and PC if neither the integrity check nor authentication check is enabled.
- Validate the TZM image type if the basic image check passed.

The beginning of the image follows the format mentioned in Table 177. The boot loader begins scanning for user images by examining the image type marker located at 0x0000 0024. If the value matches any supported image type markers, then validation of an image header will begin. After the validation of the image header is completed, the qualification continues by examining the TZM image type field.

If it is a CRC image, then the *imageLength* field value is used as the length to perform a CRC ON. See Table 177. The CRC is performed on the image in internal flash. The CRC calculation begins at offset 0x0 from the beginning of the image sector and continues up to the number of bytes specified by the length. The length does not include the *offsetToSpecificHeader* field that make up the CRC value field, which means that the CRC calculated skips the CRC value field. The result is then compared to the *offsetToSpecificHeader* entry in the structure and the image is considered valid if a match exists, otherwise the image is considered invalid. CRC is not performed if the image is not a CRC image.

If it is a signed image, then the *imageLength* field value is used as the length to perform an authentication on. The authentication will be performed on the image in internal flash. The authentication begins at the offset 0x0 from the beginning of the image sector and continues up to the number of bytes specified by the length. The *offsetToSpecificHeader* field value points to the offset that holds the certificates.

Table 177. Image header for the LPC55S6x/LPC55S2x/LPC552x device

Offset	Size in bytes	Symbol	Description
0x00	4	Initial SP	Stack pointer.
0x04	4	Initial PC	The application first execution instruction.
0x08	24	Vector table	Cortex-M33 Vector table entries .
0x20	4	imageLength	The length of current image Set to 0 if the image type is 0 as well Set to actual image length if the image type is other value.
0x24	4	imageType	Image Type 0x0000 – Normal image for unsecure boot 0x0001 – Plain signed Image 0x0002 – Plain CRC Image 0x0004 – Plain signed XIP Image 0x0005 – Plain CRC XIP Image 0x8001 – Signed plain Image with KeyStore Included
0x28	4	offsetToSpecificHeader	Offset to specific header It means offset to certificate block header if the image type is 0x01, 0x04, or 0x8001 It means the crcChecksum if the image type is 0x02 or 0x05.
0x2C	8	Vector table	Cortex-M33 Vector table entries
0x34	4	imageExecutionAddress	The execution address of the image Set to 0 if image type is 0 Set to actual image execution address if the image type is other value
0x38	8	Vector table	Cortex-M33 Vector table entries

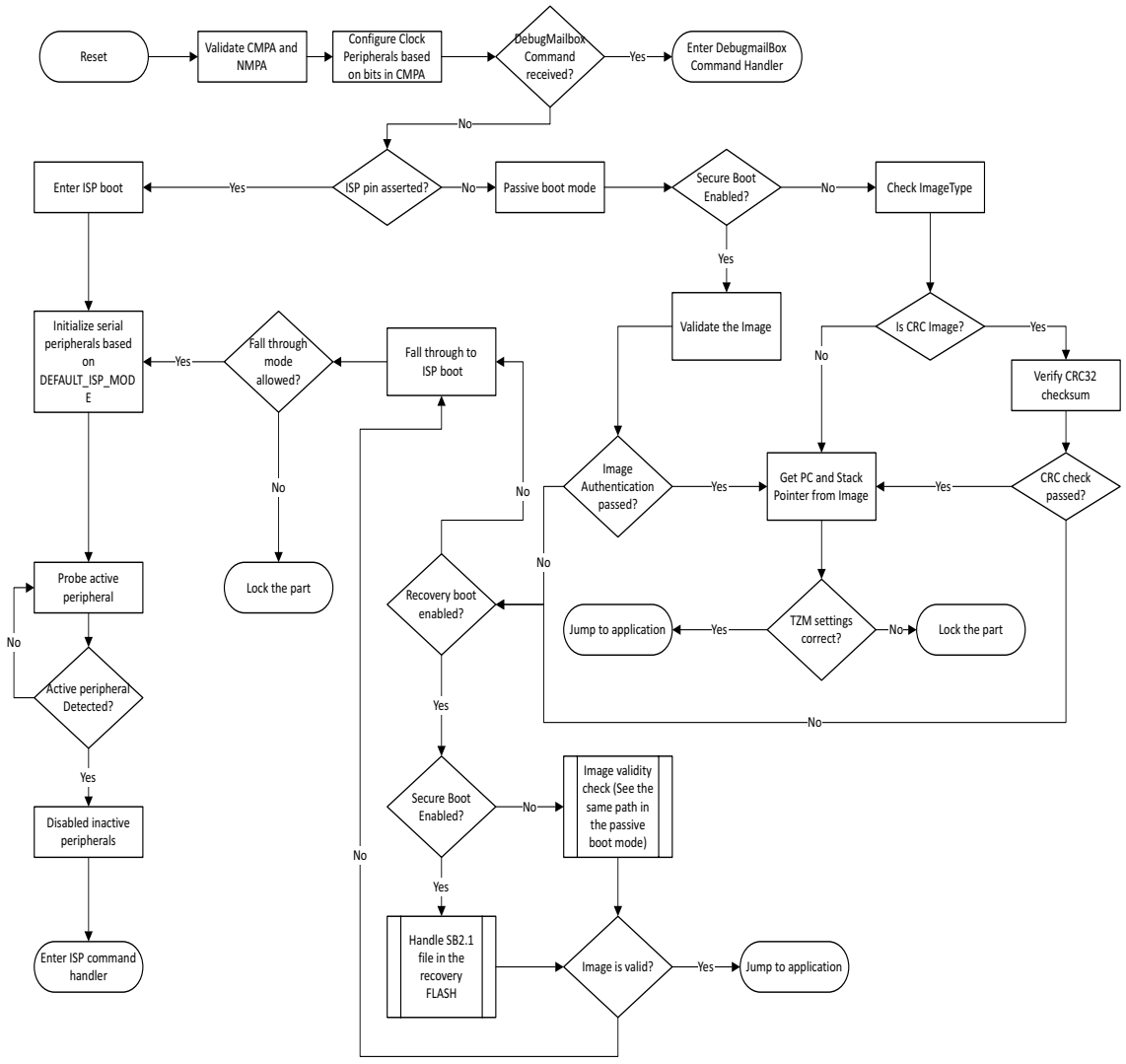


Fig 11. LPC55S6x/LPC55S2x/LPC552x boot flow chart for 1B

There are several paths into the ISP mode:

- Entry through the ISP pin assertion.
  - This mechanism is mainly used on EVK boards and customer boards where switches and jumpers are affordable in-terms of space and cost.
  - This method is used to recover a part programmed with corrupted image which is not detectable by ROM. Once in ISP mode commands are provided to re-program the flash with proper image.
- Entry through fall-through mode.
  - When the user image in flash is blank or invalid the ROM enters into ISP mode.
  - This mode of entry allows programming factory fresh parts through ISP interface.
- Entry through ROM API call.
  - Used by application code to implement in field update using ISP mechanism.

- Entry through the Debug Mailbox command.
  - This mechanism is mainly used to recover a part programmed with a corrupted image. For instance, on boards/products where the ISP pin mechanism in not affordable and SED pins are accessible or when ISP mode is disabled through the “Default ISP mode” field in the CMPA_BOOT_CFG configuration word.
  - For security sensitive application option to control access to this mechanism is provided through Debug Authentication procedure.

6.4 Boot modes

The boot modes include:

- Section 3.4.1 “Passive boot mode”.
- Section 6.4.2 “ISP boot mode”

6.4.1 Passive boot mode

The CPU clock is set to the boot speed specified in CMPA field and will boot directly from internal flash based on the image header. See Figure 11.

6.4.2 ISP boot mode

ISP boot mode can be entered as a result of failed internal flash verification or if ISP pin forces the device into ISP mode.The ISP mode is mainly used for:

- Downloading the image (initial or updated) into the internal flash from the Host.
- Provisioning the device into production life-cycle (configure secure mode, program key data, ISP fall-through mode, lock settings).

See Chapter 8 “LPC55S6x/LPC55S2x/LPC552x ISP and IAP”.

6.4.3 SPI flash recovery

For silicon revision 1B and beyond, support is provided for a recovery boot from an external 1-bit SPI flash device where an SB2.1 image is stored. The SB2.1 file is an encrypted and signed command script file which supports programming flash, PFR and other configuration commands. This feature can be implemented during OTA in the following ways:

- Recovery media model: where an external SPI flash is used to store a factory image in SB2.1 format. When the image on main flash is corrupted, ROM will attempt to recover the device by booting/executing the SB2.1 file present on the external flash device.

See: Section 7.3.6 “ROM firmware update using SB file” for more details regarding the SB2.1 file format.

In passive boot mode (ISP pin not asserted), if the internal flash image is deemed invalid, the device checks the SPI_RECOVERY_BOOT_EN (bits 3:0) in protected flash SPI_FLASH_CFG (0x9E404) to determine if SPI flash recovery is enabled. If SPI flash recovery is enabled, the boot ROM tests SEC_BOOT_EN (bits 31:30) in protected flash SECURE_BOOT_CFG (0x9E41C).

If SEC_BOOT_EN is non-zero, then the image can be booted into internal SRAM in a non-reserved region. The following commands are available for SB file recovery mode:

```
#define SBLOADER_V2_CMD_SET_IN_REC_MODE \
    ((1u << ROM_NOP_CMD) | (1u << ROM_LOAD_CMD) | (1u << ROM_JUMP_CMD) | (1u <<
     ROM_FW_VER_CHK))
```

If SEC_BOOT_EN is zero, the LPC55xx boots a plain text image with the boot address and image length specified in the image header. For plain text SPI flash recovery, the image can only be booted into internal RAM in a non-reserved region. To determine reserved regions of RAM, use the following command in ISP mode:

blhost -p COM3 get-property 12

See: Section 8.6.11.2 “1-bit SPI NOR FLASH support (for version 1B only)” for more details.

6.5 PFR region definitions

The PFR region is used as the persistent storage for the secure boot and the SoC specific parameters. It starts at fixed address 0x9DE00. Please see: Chapter 9 “LPC55S6x/LPC55S2x/LPC552x Flash API”) Protected Flash Region for details.

Table 178. Image header for the LPC55S6x/LPC55S2x/LPC552x devices

Region	Field	Description
0x9DE00-0x9E3FF	Customer In-field Programmable Area(CFPA)	See the Secure Boot chapter for more details.
0x9E400-0x9EBFF	Customer Manufacturing/Factory Programmable Area (CMPA) and Key Store Area (KSA)	
0x9EC00-0x9FDFF	NXP Manufacturing Programmed Area(NMPA)	Reserved for NXP internal use only

7.1 How to read this chapter

This document describes the Secure Boot ROM architecture for LP55S6x/LPC55S2x series.

The Secure part of ROM boot loader provides following basic operations:

- Secure boot
- Secure firmware update
- Security related miscellaneous functions

The ROM bootloader provides an API to allow integration of loader operations into customer applications.

7.2 Function description

7.2.1 Secure Boot

Secure boot provides guarantee that unauthorized code cannot be executed on a given product. It involves the device's ROM always executing when coming out of reset. The ROM will then examine the first user executable image resident in internal flash memory to determine the authenticity of that code. If the code is authentic, then control is transferred to it. This establishes a chain of trusted code from the ROM to the user boot code. This chain can be further extended, as described below.

The method used in this architecture to verify the authenticity of the boot code is to verify RSA signatures over the code. The boot code is signed with RSA private keys. The corresponding RSA public keys used for signature verification are contained in X.509 certificates that are contained in the signed image. Device supports up to four Root of Trust keys.

Device could be configured to boot plain images during development. In such case ROM does not check image to be booted, or perform only CRC32 checking, depending on configuration.

7.2.2 Secure firmware update

If firmware updates are to be performed in the field when secure boot is enabled, then a secure firmware update mechanism is preferred. Otherwise inauthentic firmware may be written to the device, causing it to not boot. In the most basic sense, secure firmware update simply performs an authentication of the new firmware prior to committing it to memory. In this case, the chain of trust is extended from the old, currently executing, code to the new code.

Another use case for secure firmware update is to hide the application binary code during transit over public media such as the web. This is accomplished by encrypting the firmware update image. As the new firmware is written into device memory, it is decrypted.

In this architecture, both cases of secure firmware update are supported. The SB file format is encrypted and digitally signed. SB file can be loaded via secure interfaces such as USB, UART, etc. or can be provided to ROM API as complete binary file. See: Section 6.4.3 “SPI flash recovery” for details.

7.2.3 Extending the chain of trust

Once secure boot has transferred CPU control to user code, that code may need to load additional pieces of code. This establishes another link in the chain of trust. The process can continue for any many nested sub-modules are required, with each parent code module authenticating the chain. Another use case is to authenticate boot code for one or more secondary CPU cores prior to releasing them from reset.

The loader API is used from customer code to verify signatures on the additional code images. Using the API to verify signatures gives complete control to the customer code over what additional code must be signed and how that code is organized in memory.

7.2.4 Miscellaneous functions

Rom provides support for various security related additional functionalities. The main are:

- Support for the load of TrustZone-M pre-configuration during ROM secure boot
- Support of booting from encrypted internal Flash regions using PRINCE peripheral module
- Debug Authentication

7.2.5 Boot flow diagram

Booting of the device is controlled by setting written in PFR (Protected Flash Region) of internal device flash memory and based on ISP pin setting.

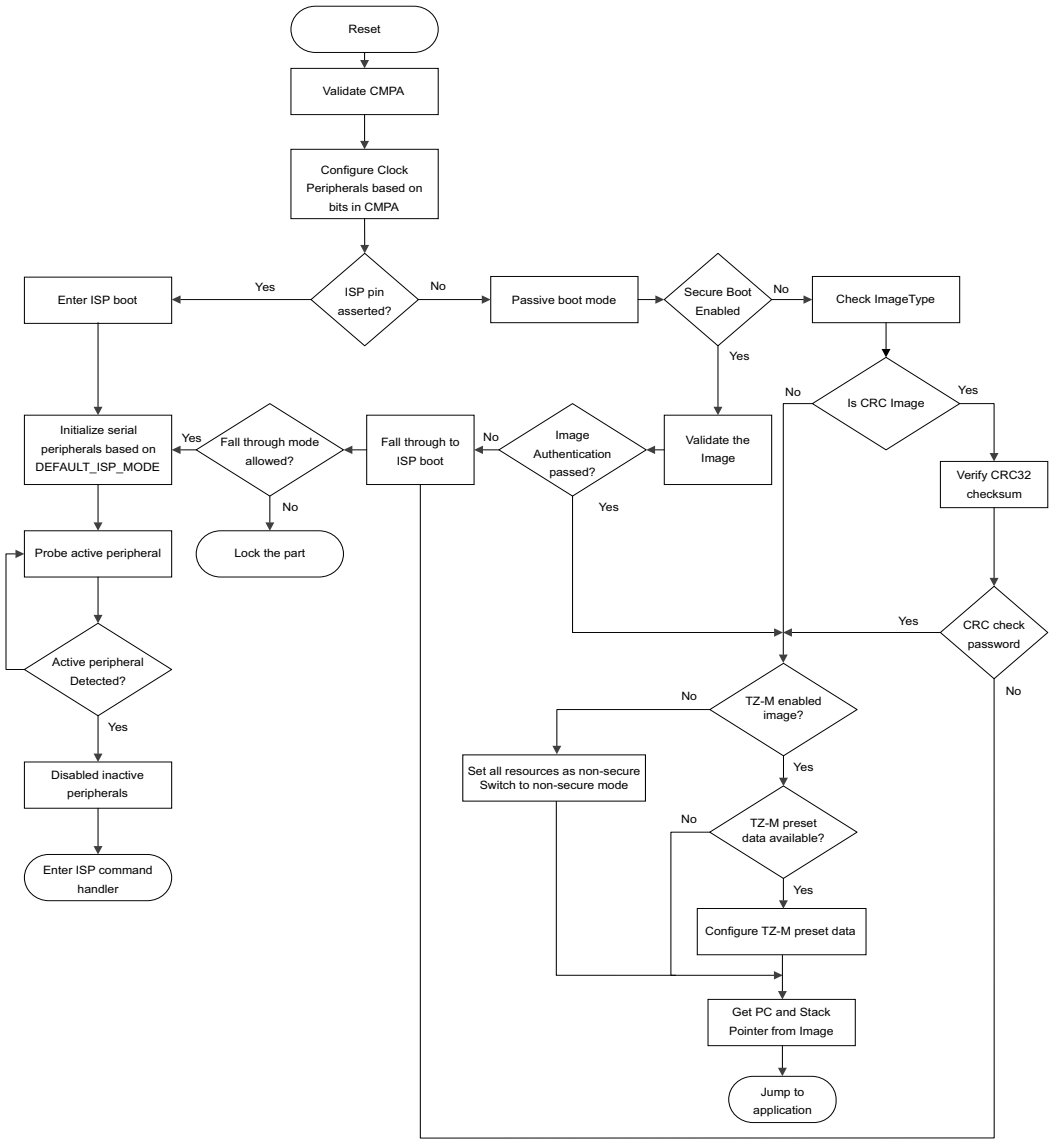


Fig 12. Secure Boot ROM Flow chart

7.2.6 Data structures

7.2.6.1 Overview

LPC55S6x/LPC55S2x/LPC552x stores configuration and PUF key store for the boot ROM in Protected Flash Region (PFR). It resides at the end of flash region and can be programmed through ROM in ISP mode.

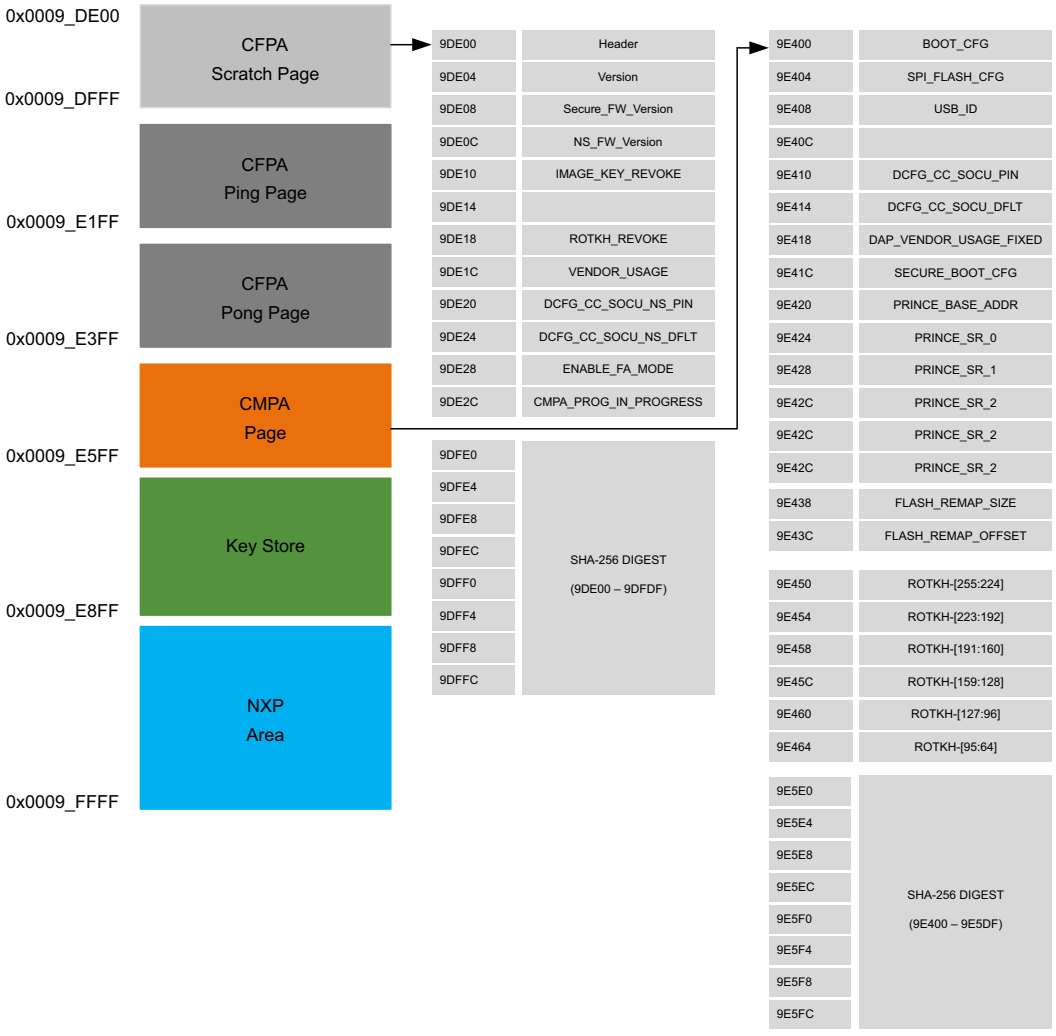


Fig 13. Protected Flash Region

7.2.6.2 Key storage in Protected Flash Region

LPC55S6x/LPC55S2x/LPC552x uses PUF controller for key wrapping. The PUF key store occupies three flash pages (1 536 bytes in total) of PFR and consists of Activation Code and six Key Codes and is managed and used mainly by the ROM during the boot and SB file processing. The key store data structure can be created during key provisioning process and written to PFR using *write to non-volatile* blhost command. The content of key storage is also available to user application by using **PFR_KeystoreGetAC** and **PFR_KeystoreGetKC** ROM API functions. See ROM API chapter for more info. During the startup, the ROM checks if valid key store data structure is present in PFR. If so, the whole key store data structure is loaded into RAM and ROM issues PUF start procedure, which initializes PUF and loads the activation code so that each key can be used if needed.

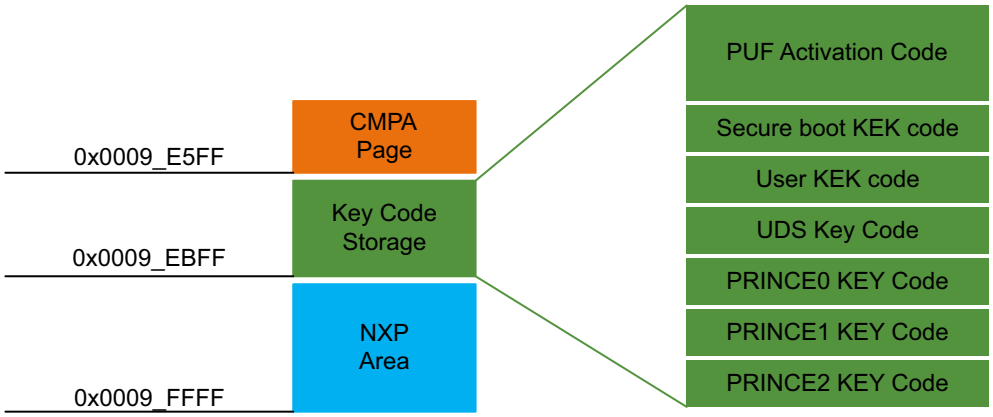


Fig 14. KeyStore area in PFR

Table 179. PUF key code storage area structure

Address	Size (bytes)	Name	Description
0x9E600	4	Key Store Header	Marker. A value of 0x95959595 means that Activation code is valid.
0x9E604	4	PUF Discharge time	Time in milliseconds to wait until PUF SRAM fully discharges. Only effective when PUF Start fails. Set to zero to use default discharge time.
0x9E608	1192	Activation Code	Device specific PUF activation code generated by enroll command during key provisioning.
0x9EAB0	4	SBKEK Key Code Header	Marker. A value of 0x59595959 means that key code is valid.
0x9EAB4	52	SBKEK Key Code	Key Code for wrapped SBKEK key.
0x9EAE8	4	USERKEK Key Code Header	Marker. A value of 0x59595959 means that key code is valid.
0x9EABC	52	USERKEK Key Code	Key Code for wrapped USERKEK key.
0x9EB20	56	-	Reserved.
0x9EB58	4	PRINCE Region 0 Key Code Header	Marker. A value of 0x59595959 means that key code is valid.
0x9EB5C	52	PRINCE Region 0 Key Code	Key Code for wrapped PRINCE Region 0 key.
0x9EB90	4	PRINCE Region 1 Key Code Header	Marker. A value of 0x59595959 means that key code is valid.
0x9EB94	52	PRINCE Region 1 Key Code	Key Code for wrapped PRINCE Region 1 key.
0x9EBC8	4	PRINCE Region 2 Key Code Header	Marker. A value of 0x59595959 means that key code is valid.
0x9EBCC	52	PRINCE Region 2 Key Code	Key Code for wrapped PRINCE Region 2 key.

7.3 Keys

This chapter will describe purpose of individual keys used by the bootloader. Key Codes for given keys are generated (wrapped) using PUF by data supplied to blhost by key provisioning commands with specified key type and key length. Keys stored with PUF key

index 0 can be unwrapped only on secure key hardware bus so that only security peripherals connected to this bus are able to use this code. See Chapter 48 "LPC55S6x/LPC55S2x/LPC552x Security features" for more details about how to generate and load the key store into device PFR.

7.3.1 PUF key code format

Table 180. PUF key code format

Offset	Bytes	Name	Description
0x0	4	Type, Index, Size	Bits 1:0 – Type 2'b00: User key 2'b01: Generate key 2'b10: Invalid 2'b11: Invalid Bits 7:2 – Reserved Always set to 0 Bits 11:8 - Key Index PUF key index 0 - 15 Bits 23:12 – Reserved Always set to 0 Bits 29:24 – Key Size Size of the key (64-bit multiple) Bits 31:30 – Reserved Always set to 0
0x4	48	Key Code	Wrapped key code data

7.3.2 Key descriptions

SBKEK– Secure Binary Key Encryption Key

- Used for SB2 firmware update image decrypt
- AES-256 symmetric key
- blhost key type 3
- PUF key index 0

USERKEK – User Key Encryption Key

- Not used by LPC55S6x/LPC55S2x/LPC552x bootloader. Available for user as pre-shared master key
- AES-256 symmetric key
- blhost key type 11
- PUF key index 0

PRINCE Region 0-2 key

- Key used to encrypt/decrypt data in internal flash memory when PRINCE is enabled for given memory region.
- 128-bit symmetric key

- Region 0 - blhost key type 7
- Region 1 - blhost key type 8
- Region 2 - blhost key type 9
- All three keys are PUF key index 0

7.3.2.1 Secure boot related configuration fields in PFR

7.3.2.1.1 CMPA page

The CMPA (Customer Manufacturing/Factory Programmable Area) page contains settings for signed image in secure boot configuration, PRINCE configuration registers (if encrypted flash is needed) and 32 bytes of Root Key Table Hash (RKTH). Only secure boot related fields are described in this chapter.

Table 181. CMPA Secure boot configuration overview

Address	Bytes	Name	Description
0x9E41C	4	SECURE_BOOT_CFG	Secure boot configuration flags
0x9E420	4	PRINCE_BASE_ADDR	PRINCE configuration and region base addresses
0x9E424	4	PRINCE_SR_0	Region 0, sub-region enable.
0x9E428	4	PRINCE_SR_1	Region 1, sub-region enable
0x9E42C	4	PRINCE_SR_2	Region 2, sub-region enable
0x9E450	32	RKTH	Root Key Table Hash

Note: PRINCE region base addresses are shown below:

PRINCE base address: 0x4003 5000 (non-secure)

PRINCE base address for region 0: 0x4003 5018

PRINCE base address for region 1: 0x4003 5028

PRINCE base address for region 2: 0x4003 5038

SECURE_BOOT_CFG configuration word

Table 182. SECURE_BOOT_CFG word bit field definitions

Address	Bit(s)	Name	Description
0x9E41C	1:0	RSA4K	Use RSA4096 keys only 2'b00: Allow RSA2048 and higher 2'b01: RSA4096 only 2'b10: RSA4096 only 2'b11: RSA4096 only
	3:2	DICE_INC_NXP_CFG	Include NXP area in DICE computation 2'b00: not included 2'b01: included 2'b10: included 2'b11: included
	5:4	DICE_CUST_CFG	Include CFPA page and key store area in DICE computation 2'b00: not included 2'b01: included 2'b10: included 2'b11: included
	7:6	SKIP_DICE	Skip DICE computation 2'b00: Enable DICE 2'b01: Disable DICE 2'b10: Disable DICE 2'b11: Disable DICE
	9:8	TZM_IMAGE_TYPE	TrustZone-M image mode 2'b00: TZ-M image mode is taken from application image header 2'b01: TZ-M disabled image, boots to non-secure mode 2'b10: TZ-M enabled image, boots to secure mode 2'b11: TZ-M enabled image with TZ-M preset, boot to secure mode TZ-M pre-configured by data from application image header
	11:10	BLOCK_SET_KEY	Block PUF key code generation 2'b00: Allow PUF Key Code generation 2'b01: Disable PUF Key Code generation 2'b10: Disable PUF Key Code generation 2'b11: Disable PUF Key Code generation
	13:12	BLOCK_ENROLL	Block PUF enrollment 2'b00: Allow PUF enroll operation 2'b01: Disable PUF enroll operation 2'b10: Disable PUF enroll operation 2'b11: Disable PUF enroll operation
	15:14	DICE_INC_SEC_EPOCH	Include security epoch area in DICE computation 2'b00: not included 2'b01: included 2'b10: included 2'b11: included
	29:16	RESERVED	Reserved, filled with zeros
	31:30	SEC_BOOT_EN	Secure boot enable 2'b00: Plain image (internal flash with or without CRC) 2'b01: Boot signed images. (internal flash, RSA signed) 2'b10: Boot signed images. (internal flash, RSA signed) 2'b11: Boot signed images. (internal flash, RSA signed)

PRINCE_BASE_ADDR: Contains various configuration for PRINCE peripheral to be set by ROM bootloader during device startup.

Table 183. PRINCE configuration

Address	Bit(s)	Name	Description
0x9E420	3:0	ADDR0_PRG	Programmable portion of the base address of region 0.
	7:4	ADDR1_PRG	Programmable portion of the base address of region 1.
	11:8	ADDR2_PRG	Programmable portion of the base address of region 2.
	15:12	RESERVED	Should be filled with zeros.
	17:16	RESERVED	Should be filled with zeros.
	19:18	LOCK_REG0	Lock PRINCE region 0 settings. 2'b00: Region is not locked 2'b01: Region is locked 2'b10: Region is locked 2'b11: Region is locked
	21:20	LOCK_REG1	Lock PRINCE region 1 settings. 2'b00: Region is not locked 2'b01: Region is locked 2'b10: Region is locked 2'b11: Region is locked
	23:22	RESERVED	Should be filled with zeros.
	25:24	REG0_ERASE_CHECK_EN	PRINCE region 0 enable checking whether all encrypted pages are erased together. 2'b00: Region is disabled 2'b01: Region is enabled 2'b10: Region is enabled 2'b11: Region is enabled
	27:26	REG1_ERASE_CHECK_EN	PRINCE region 1 enable checking whether all encrypted pages are erased together. 2'b00: Region is disabled 2'b01: Region is enabled 2'b10: Region is enabled 2'b11: Region is enabled
	29:28	REG2_ERASE_CHECK_EN	PRINCE region 2 enable checking whether all encrypted pages are erased together. 2'b00: Region is disabled 2'b01: Region is enabled 2'b10: Region is enabled 2'b11: Region is enabled
	31:30	RESERVED	Should be filled with zeros.

PRINCE_SR_x: When on-the-fly encryption/decryption of internal flash using PRINCE is enabled, ROM configures sub-region enable bits for given memory region according to value stored in this word

Table 184. PRINCE sub-region enable bits

Address	Bit(s)	Name	Description
0x9E424	31:0	SRn_EN	Each bit in this field enables a sub-region of crypto region x at offset 8kB*n, where n is the bit number. A 0 in bit n bit means encryption and decryption of data associated with sub-region n is disabled. A 1 in bit n means that data written to sub-region n during flash programming when ENC_ENABLE.EN = 1 will be encrypted, and flash reads from sub-region n will be decrypted using the PRINCE.

RKTH: RKTH is 32 byte SHA-256 hash of SHA-256 hashes of up to four root public keys. Multiple root public keys are supported to allow for key revocation.
The structure of this table is shown here

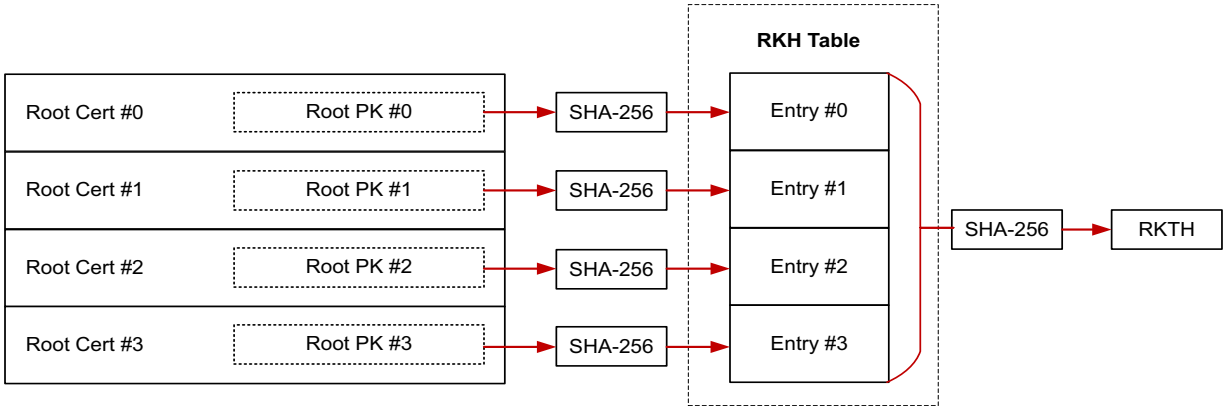


Fig 15. RKTH generation process

Each entry in the table is a SHA-256 computed over the concatenation of an RSA public key's modulus and exponent (modulus || exponent). Both modulus and exponent must be in big endian byte order, with the minimum number of bytes required to represent the value. For instance, an exponent of 65537 would be represented by a 3-byte value of [01 00 01], while an exponent of 3 would be represented by a single byte of that value. The entire RKH table is itself hashed with SHA-256. This final hash is then stored in the RKTH field in PFR.

For i in 0...3:

Let M i = BE(Modulus i) Let E i = BE(Exponent i)

Let RKH i = SHA256(M i || E i)

Let RKTH = SHA256(RKH 0 || RKH 1 || RKH 2 || RKH 3)

The number of hashes of keys in the RKH table must be from at least 1 through a maximum of 4. Unused table entries must be set to all 0 bytes. When searching the RKH table for a key's hash, the loader will stop at the first entry that is all zeroes.

The extra root public keys and root certificates must be created in advance and would be held in reserve in case a public key had to be revoked. The customer is responsible for implementing the mechanism to determine whether a key needs to be revoked, and then set the appropriate `RKTH_REVOKE` bit(s). This would usually be done through an authenticated connection with a server during a firmware update.

Note: Only one of the root certificates whose keys are listed in the RKH table may be included in the certificate table at a time.

Table 185. RKTH layout in CMPA

Address	Description	Address	Description
0x9E450	RKTH[255:224]	0x9E460	RKTH [127:96]
0x9E454	RKTH [223:192]	0x9E464	RKTH [95:64]
0x9E458	RKTH [191:160]	0x9E468	RKTH [63:32]
0x9E45C	RKTH [159:128]	0x9E46C	RKTH [31:0]

7.3.2.1.2 CFPA page

The CFPA (Customer Field Programmable Area) page contains three monotonic counters, RKTH revocation fields and storage for three PRINCE region IV codes. Only secure boot related fields are described in this chapter.

Table 186. CFPA page layout

Address	Byte(s)	Name	Description
0x9DE08	4	Secure_FW_Version	Secure firmware version (Monotonic counter)
0x9DE0C	4	NS_FW_Version	Non Secure firmware version (Monotonic counter)
0x9DE10	4	IMAGE_KEY_REVOKE	Image key revocation ID (Monotonic counter)
0x9DE18	1	RKTH_REVOKE	Used for revocation of individual Root keys
0x9DE30	56	PRINCE Region 0 IV CODE	IV code used for PRINCE region 0
0x9DE68	56	PRINCE Region 1 IV CODE	IV code used for PRINCE region 1
0x9DEA0	56	PRINCE Region 2 IV CODE	IV code used for PRINCE region 2

Secure_FW_version: Used during SB2 file loading. The value written in this configuration word must be always lower or equal to secure FW version specified in elftosb .bd file used for creating SB2 file. Otherwise the SB file load will be rejected.

NS_FW_version: Used during SB2 file loading. The value written in this configuration word must be always lower or equal to non-secure FW version specified in elftosb .bd file used for creating SB2 file. Otherwise the SB file load will be rejected.

IMAGE_KEY_REVOKE: This value is checked during image authentication process. The x509 serial number field in the image signing certificate is used the following way: byte 0 shall be 0x3c, byte 1 shall be 0xc3, byte 2 and byte 3 form an unsigned 16-bit integer whose value is compared with the IMAGE_KEY_REVOKE value in the PFR. On mismatch, the image authentication process will fail. Only 17 revocation IDs are possible. (0x0, 0x1, 0x3, 0x7, 0xF, 0x1F, 0x3F, 0x7F, 0xFF … 0xFFFF). One bit should be set on every revocation starting from lower bit 0 to 16:

0b0 ->0b1 -> 0b11->0b111 .....

To avoid bricking the device if power loss happens after FW update but before IMAGE_KEY_REVOKE is updated, LPC55Sxx boot ROM allows a roll forward (only by 1), and cannot be rolled back.

RKTH_REVOKE: Each of four RoT Keys can be revoked. When trying to boot Images that are signed using revoked RoT key they will be rejected during the authentication process and will fail to boot if SEC_BOOT_EN is set to boot only signed images.

Table 187. RKTH table bit field description

Address	Bit(s)	Name	Description
0x9DE18	1:0	RoTK0_EN	RoT Key 0 enable 2'b00: Invalid 2'b01: RoT Key 0 is enabled 2'b10: RoT Key 0 is revoked 2'b11: RoT Key 0 is revoked
	3:2	RoTK1_EN	RoT Key 1 enable 2'b00: Invalid 2'b01: RoT Key 1 is enabled 2'b10: RoT Key 1 is revoked 2'b11: RoT Key 1 is revoked
	5:4	RoTK2_EN	RoT Key 2 enable 2'b00: Invalid 2'b01: RoT Key 2 is enabled 2'b10: RoT Key 2 is revoked 2'b11: RoT Key 2 is revoked
	7:6	RoTK3_EN	RoT Key 3 enable 2'b00: Invalid 2'b01: RoT Key3 iis enabled 2'b10: RoT Key3 is revoked 2'b11: RoT Key 3 is revoked
	31:8	RESERVED	Should be filled with zeros

PRINCE region x IV code: Initial vector value for PRINCE region x in PUF Key Code format. This value is used to configure IV for PRINCE regions during ROM startup. It is generated and used only by bootloader. This value should not be modified by the user.

7.3.3 Plain image structure

Unsigned Plain CRC images are supported by non-Secure versions of LPC55S6x/LPC55S2x/LPC552x as well as Secure versions during development life-cycle state of S parts (LPC55S6x/LPC55S2x/LPC552x).

The structure of unsigned CRC images is shown here:

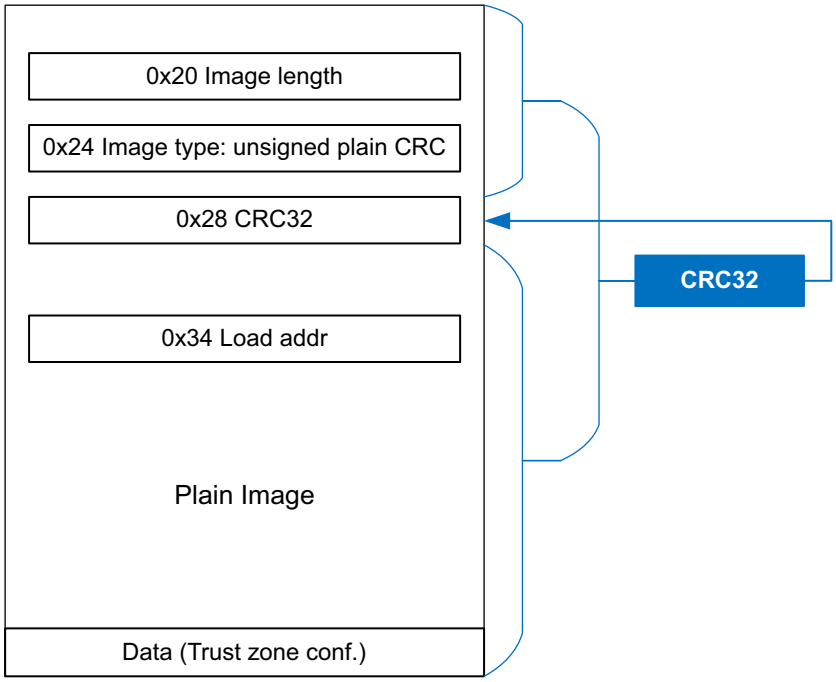


Fig 16. Structure of Unsigned CRC images

7.3.4 Signed image structure

Images are signed using the RSASSA-PKCS1-v1_5 algorithm. The digest is computed using SHA-256, 2048-bit, or 4096-bit RSA keys are supported.

The structure of signed images is shown here:

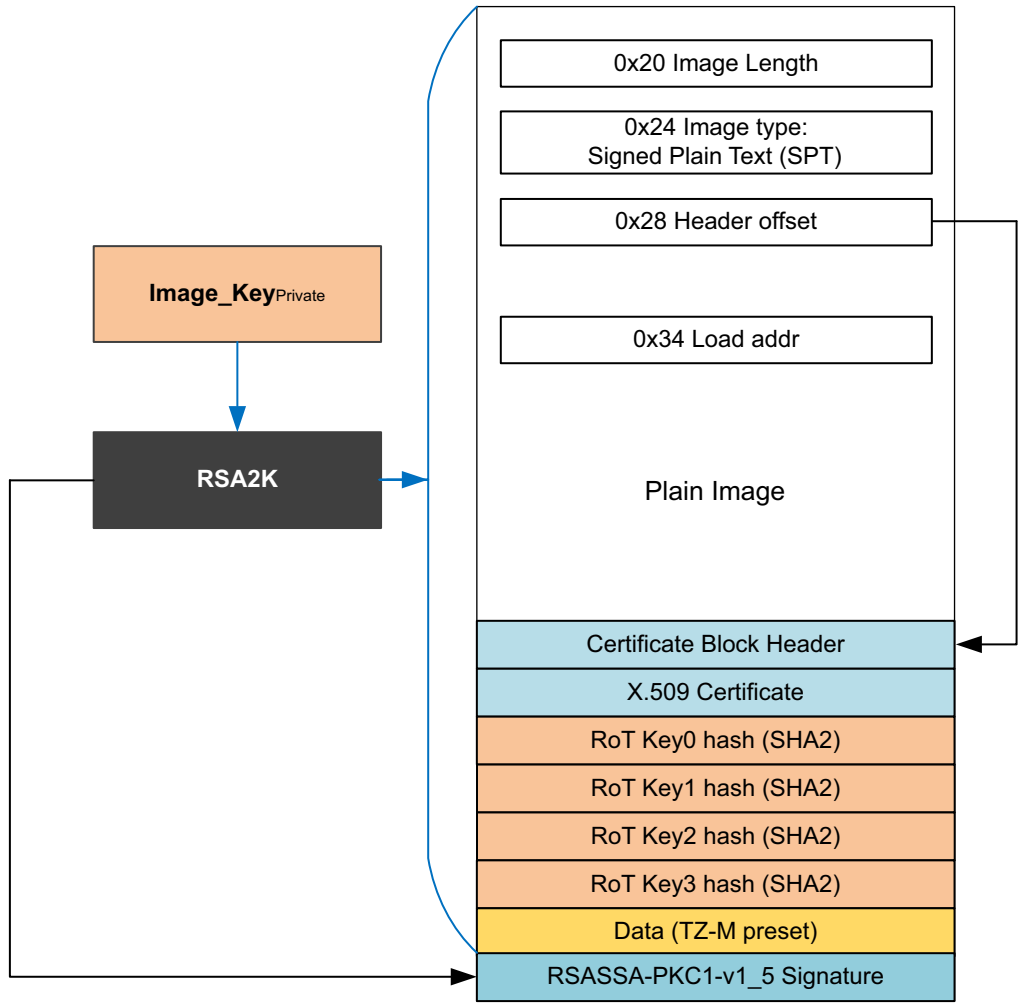


Fig 17. Structure of Signed Images

Image length - total length of the image in bytes including signature

Image type -

Table 188. LPC55S6x/LPC55S2x/LPC552x Image Type (word at offset 0x24)

31:16	Reserved	Set to 0
15	Reserved	Set to 0
14	TZ-M Image Type	0: TZ-M enabled image. The image uses TZ-M 1: TZ-M disabled image. The image doesn't uses TZ-M

Table 188. LPC55S6x/LPC55S2x/LPC552x Image Type (word at offset 0x24)

13	TZ-M Preset	0: No TZ-M peripherals preset 1: TZ-M peripherals preset. The TZ-M related peripherals are configured by bootloader based on data appended to an image (after RoT Key Hash table))
12:8	Reserved	Set to 0
7:0	Image Type	0x0: plain image 0x4: Internal flash, plain, signed 0x5: Internal flash, plain, CRC Other values are reserved

Header Offset - A 32-bit offset from the beginning of the signed image to the certificate block header, called offsetToCertificateBlockInBytes, must reside at offset 0x28 from the start of the signed image. An executable code image in internal flash is expected to start with an NVIC vector table. The word at offset 0x28 is a reserved slot in the vector table.

As an example, if an image resides in flash at a non-zero address (say 0x8000), and its certificate block header is at address 0x24000, then the word at 0x8028 will contain the value 0x1c000.

Here is a standard Cortex-M33 NVIC vector table with the offset to the certificate block header highlighted.

Table 189.

Offset	Usage
0x00	Initial SP
0x04	Reset
0x08	NMI
0x0C	HardFault
0x010	MemManage
0x014	BusFault
0x018	UsageFault
0x01C	*Reserved*
0x020	Image Length
0x024	Image Type
0x028	**offsetToCertificateBlockInBytes**
0x030	SVC
0x034	DebugMon
0x038	*Reserved*
0x03C	SysTick

7.3.5 Certificate block

The certificate block consists of the certificate block header, the certificate table, and the RKH table concatenated together.

The certificate block can reside anywhere within the signed image, but must be fully contained within the signed data, such that the certificate block itself is signed. The most common constructions will have the certificate block placed at either the beginning (after the vector table) or end of the signed data.

The structure of the certificate block looks like this:

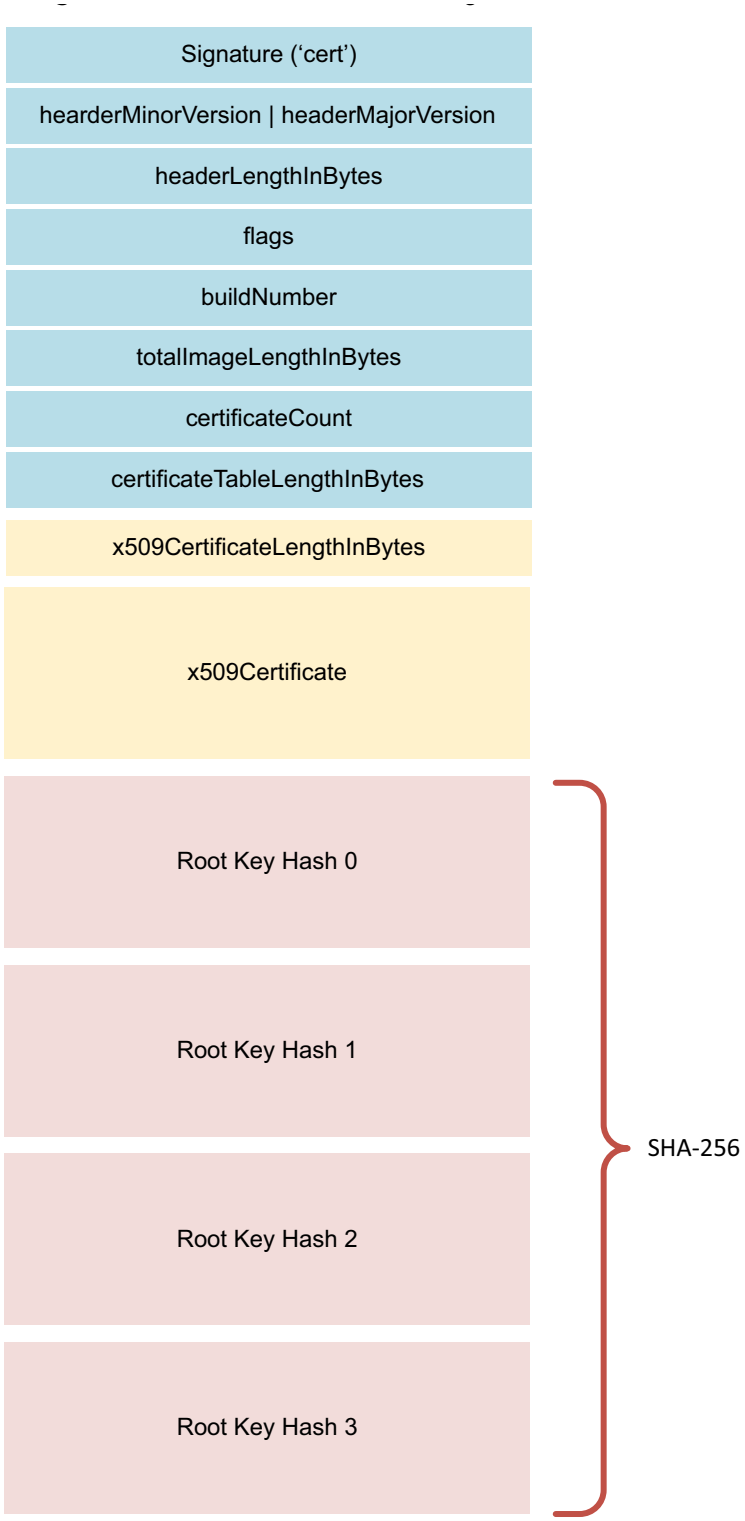


Fig 18. Structure of Certificate Block

7.3.5.1 Certificate block header

The certificate block header (or just certificate header) is a structure containing information required to properly verify a signed image. As described above, it is pointed to by the `offsetToCertificateBlockInBytes` header offset field.

`Let O header = (offsetToCertificateBlockInBytes)` The first word of the certificate block header must be 4-byte aligned.

Descriptions of the fields in the certificate block header:

Table 190.

Field	Description
signature	Always set to 'cert'.
headerMajorVersion	Set to 1
headerMinorVersion	Set to 0
headerLengthInBytes	Number of bytes long the header is, starting from the signature. Does not include the certificate table
flags	Reserved for future use
buildNumber	User specified build number for the signed image. Allows user to prevent reverting to old versions. The API compares this against the minBuildNumber specified in kb_options_t
totalImageLengthInBytes	Length in bytes of the signed data
certificateCount	Must be greater than 0
certificateTableLengthInBytes	Total length in bytes of the certificate table

The key field in the certificate header is `totalImageLengthInBytes`. This field indicates the number of bytes of signed data, starting at offset 0 of the image. The entire certificate block **must** be contained within the signed data.

The signature field can be treated as 4-character string, without a terminating null byte, with a value of 'cert'. Represented as a 32-bit little endian constant, the value would be (('c') | ('e' << 8) | ('r' << 16) | ('t' << 24)).

7.3.5.2 Certificate table

Immediately following the certificate block header is the certificate table. It consists of a complete chain of one or more X.509 certificates, each prefixed with a length word.

Let O cert-table = (offsetToCertificateBlockInBytes + headerLengthInBytes)

The `x509CertificateLengthInBytes` field for each certificate must be set to the length of that certificate's data in bytes, rounded up to the next word (4-byte) alignment. Thus, `x509CertificateLengthInBytes` must be divisible by 4. `x509Certificate` contains the actual certificate data, and can be of variable length. There may be from 0-3 bytes of padding inserted after the certificate data. The `cert_entry` struct is repeated for `certificateCount` entries in the table. The total number of bytes occupied by the table must equal `certificateTableLengthInBytes`, and must always be divisible by 4.

There are a number of restrictions on the certificates:

Only x509 v3 format certificates are supported. Must be DER encoded.

Must use RSA-2048, RSA-3072 or RSA-4096 and SHA-256.

All certificates must use RSA keys with a modulus bit length greater than or equal the RSA bit length, specified by the security profile. This means that a 4096-bit or 3072-bit root key followed by a 2048-bit image key is allowed, if the security profile is set to 2048-bit keys.

The SHA-256 hash of the public key contained in the first certificate in the table must be present in the RKH table.

The certificate table can contain one or more certificates. Certificates must be positioned in the table starting with the root certificate, followed by each subsequent certificate in the chain in order of signing. The final certificate in the table is called the image signing certificate. Using a single certificate is allowed. In this case, the sole certificate must be self-signed and must not be a CA. If multiple certificates are used, the root must be self-signed and all but the last must be CAs.

The RSA public key from the root certificate is denoted RPK, while the RSA public key from the image signing certificate is denoted IPK. The two most common configurations will be:

One self-signed certificate

Self-signed root CA certificate, followed by image signing certificate which is itself signed by the root certificate.

7.3.6 ROM firmware update using SB file

The Secure Binary (SB) image format is a command-based firmware update image. It has a long history, and has been used on multiple STMP and i.MX devices. The 0A version of the LPC55S6xx silicon supports version 2.0 and the 1B version of the LPC55S6xx supports version 2.1 of the SB image format. These new versions update the encryption scheme to use modern algorithms. They also add support for signed images. The main difference between version 2.0 and version 2.1 is in the usage of the digital signature.

With version 2.0, introduced with the first version of LPC55S6x/LPC55S2x/LPC552x silicon, the digital signature of the SB 2.0 file is optional, the signed hash is the hash of complete SB 2.0 file and the ROM bootloader's ReceiveSBFile command never verifies the signature. SB file version 2.1, introduces with the next version of

LPC55S6x/LPC55S2x/LPC552x silicon, makes the usage of digital signature mandatory. The ReceiveSBFile command verifies the digital signature.

The SB 2.0 and 2.1 file format also uses AES encryption for confidentiality and HMAC for extending trust from the signed part of the SB file to the command and data part of the SB file. These two keys (AES decrypt key and HMAC key) are wrapped in the RFC3394 key blob, for which the key wrapping key is the SBKEK key.

The layout of an SB 2.0 and SB 2.1 file is shown in the elftosb tool User's Guide. The elftosb tool is the NXP image signing and SB file creating tool for Windows/Linux/MAC. In this chapter, we provide introductory description of the SB file format components, while for the exact description, one shall refer to the elftosb tool User's Guide.

7.3.6.1 Header

The header contains plaintext information about the SB file, such as version, length and nonce for AES CTR mode. With SB file version 2.1, the header also contains RSA

signature of the hash computed from all the header plaintext data. The RSA Verify logic implemented in the ROM for the SB file version 2.1 is the same as the logic for the internal flash image authentication, that ROM can execute during the secure boot flow.

7.3.6.2 Header MAC

Only present in SB file version 2.0.

The header MAC is a message authentication code that provides secret key based authentication of the SB file header. It is computed with the HMAC-SHA256 algorithm using a 256-bit secret key called K MAC. The HMAC is generated over the entire header plus the section table, including any required padding. It is 32 bytes in size.

7.3.6.3 Key blob

The key blob wraps two 256-bit keys using the RFC3394 algorithm. It provides integrity and authenticity over the wrapped keys. The two keys in the key blob are:

1. Data Encryption Key (K $_{DEK}$).
2. MAC Key (K $_{MAC}$)

The K $_{DEK}$ is used to encrypt section data in the SB file. K $_{MAC}$ is used for header HMAC (if available) and section HMACs.

Both keys are uniquely generated each time an SB file is built. They are wrapped with the SBKEK that is programmed into the target device's PFR.

7.3.6.4 Sections

The content of SB files is divided into an arbitrary number of sections, each with a unique ID. Every section is preceded with a boot tag that acts as a header, plus an HMAC table. A section may be either a bootable section that contains boot commands, or a data section containing data not used by the loader. There must be at least 1 bootable section for an SB file to be valid. The LPC55xx ROM loader supports only a single section.

7.3.6.4.1 Boot tag

Boot tags prefix a section with the information about that section. They form a linked list within the SB file, allowing the loader to sequentially search for a given section. Boot tags are always encrypted using AES-CTR.

7.3.6.4.2 Section MAC table

Following the boot tag is a table of MACs used to verify the integrity and authenticity of section data. These MACs are computed using the HMAC-SHA256 algorithm with the K MAC key from the SB file's key blob. The number of MACs for a section is configurable by the user to trade between memory utilization and protection granularity.

7.3.6.4.3 Bootable section

A section that has the bootable section flag set is called a bootable section. It contains a sequence of boot commands that are processed by the loader to perform a firmware update.

The boot commands are described in the elftosb User’s Guide. The LPC55xx ROM loader provides the support for the following bootloader commands:

WriteMemory, FillMemory, ConfigureMemory, FlashEraseAll, FlashEraseRegion,

The WriteMemory and FillMemory commands can be used to write data to RAMs. WriteMemory can be also used to program internal flash, including the PFR CFPA page, assuming the flash is erased, for example, by FlashEraseAll or FlashEraseRegion commands. ConfigureMemory command can be used to configure LPC55xx PRINCE on-the-fly encryption module.

SB 2.1 introduces two new commands that can be used to prevent firmware roll-back:

SecureFirmwareVersion

NonsecureFirmwareVersion

The recovery boot mode on the 1B version of the LPC55S6xx that is using SB 2.1, supports the following commands:

- ROM_NOP_CMD
- ROM_LOAD_CMD
- ROM_JUMP_CMD
- ROM_FW_VER_CHK_CMD

7.3.6.4.4 Data section

Any section for which the bootable section flag is cleared is a data section. The loader does not examine the contents of such sections; it simply skips over them. Data sections may optionally be unencrypted by setting the cleartext flag. The SB 2.0 optional certificate block header is an example of a data section.

7.3.6.4.5 Certificate block header, certificates and RKH table

A signed SB file must have a certificate block header. For SB 2.0, the certificate header in a signed SB file is stored in a data section with a tag of 'sign'. This section must have its cleartext flag set to mark it as unencrypted. It also must have the bootable flag cleared to cause the SB loader to skip the certificate header when executing the SB file. The certificate chain and RKH table are also included in the 'sign' section, since they must always immediately follow the certificate block header. The 'sign' section itself may be at any position within the sequence of sections, though usually would be the first or last.

For SB 2.1, the certificate block header, certificate chain and RKH table is mandatory part of the SB 2.1 file header.

7.3.6.4.6 Signature

An SB 2.0 file may optionally be signed. If signed, a signature in RSASSA-PKCS1-v1_5 format is appended to the end of the file. The digest is computed over the entire SB 2.0 file except for the signature itself.

For SB 2.1, the signature is mandatory and immediately follows the RKH table.

The SB 2.0 file and the SB 2.1 file header have the same structure as a signed image in the internal flash. Thus, the ROM’s image authenticate function is used to verify digital signature of internal flash image, SB 2.0 signed file and SB 2.1 header. The signature in RSASSA-PKCS1-v1_5 format is appended to the tail end of the internal flash image / SB 2.0 file and to the tail end of the SB 2.1 header.

7.3.6.5 Usage of firmware update

SB 2.0 files are always encrypted and optionally signed. SB 2.1 files are always encrypted, and the header is always signed. Users may call the loader API from their application code to authenticate an image, either signed code or an SB file.

The recommended method to perform secure firmware updates is as follows:

- User application receives an encrypted SB file containing new firmware and stores it in external SPI flash, or a similar memory.
- Use API to authenticate SB file.
- Use API to decrypt and load the SB file.
- If also using secure boot, the API can be used to authenticate the new firmware in flash before rebooting into it. If this final authentication fails, the new firmware should be made non-executable by erasing and writing over critical regions of it such as the vector table. Even if not using secure boot, the code written to flash can still be signed to support this final authentication step.

7.3.6.5.1 Device setup required for SB file 2.0 processing

The SB 2.0 processing by ROM depends on the presence of a valid key store setup with SBKEK key code. Below is an example of such key store provisioning into the device using the ROM bootloader key provisioning commands:

:: PUF enroll (generate activation code into key store)

blhost -p com6 -- key-provisioning enroll

:: install SBKEK into key store. SBKEK key type = 3.

blhost -p com6 -- key-provisioning set_user_key 3 sbkek.bin

:: install USERKEK into key store. USERKEK key type = 11.

blhost -p com6 -- key-provisioning set_user_key 11 userkek.bin

:: generate random PRINCE region 0. PRINCE region 0 key type = 7

blhost -p com6 -- key-provisioning set_key 7 16

:: generate random PRINCE region 1. PRINCE region 1 key type = 8

blhost -p com6 -- key-provisioning set_key 8 16

:: generate random PRINCE region 2. PRINCE region 0 key type = 9

blhost -p com6 -- key-provisioning set_key 9 16

:: save the key store into PFR

blhost -p com6 -- key-provisioning write_key_nonvolatile 0

7.3.6.5.2 Device setup required for SB file 2.1 processing

The SB 2.1 processing by ROM depends on the presence of a valid key store with SBKEK key code. This is the same as with SB 2.0.

The SB 2.1 processing by ROM also depends on the presence of RKT hash (RKTH) in the CMPA page in the PFR.

7.3.6.6 Secure ROM API

The ROM API table is located at address *0x130010f0* and contains absolute ROM API function addresses which can be called using function pointers. PRINCE ROM API section starts at address *0x1300119C* and *skboot_authenticate()* function address is located at *0x1300120C*. Only secure boot related functions are described in this chapter. See LPC55S6x/LPC55S2x/LPC552x User Manual for complete list of ROM API functions.

The main purpose of these APIs is to provide access to functions used and implemented in ROM to authenticate the application image and to configure PRINCE on-the-fly encryption/decryption peripheral.

Table 191. Secure ROM API summary

Address in ROM API table	Absolute function address	Function
0x1300120C	0x1300A34F	skboot_status_t skboot_authenticate(const uint8_t *imageStartAddr, secure_bool_t *isSignVerified)
0x13001210	0x13003717	void skboot_hashcrypt_irq_handler(void)
0x1300119C	0x13006B3B	skboot_status_t bus_crypto_engine_gen_new_iv(uint32_t region, uint8_t *iv_code, secure_bool_t store, flash_config_t *flash_context)
0x130011A0	0x13006C6D	skboot_status_t bus_crypto_engine_load_iv(uint32_t region, uint8_t *iv_code)
0x130011A4	0x13006955	skboot_status_t bus_crypto_engine_set_encrypt_for_address_range

7.4 Image authentication API

7.4.1 skboot_authenticate

This API function can be used to verify authenticity of an image. The ROM uses this function during the secure boot flow to authenticate an image in the internal flash, and it also uses it to verify authenticity of the SB 2.0/2.1 files. If a user application calls skboot_authenticate() directly or indirectly from SB file processing functions kb_init/kb_process/kb_deinit, the user HASH interrupt vector shall call the HASH_IRQHandler() function for handling of the Hash-crypt IP interrupt. This is due to the fact that the hashing is implemented as non-blocking for shorter computation time – while the Hash-crypt AHB master fetches data for hashing, the CPU and Casper co-processor work on RSA Verify.

It is important to note that the skboot_authenticate() ROM function uses global variables in RAM.

Thus, the caller has to assure that it doesn't have any data in the global variables location before the function call. The caller shall discard the data in the global variables location after the function returns.

The ROM reserved space for global variables in RAM on this LPC55S6x/LPC55S2x/LPC552x product is:

0x30000000 to 0x30003FFF

0x14005000 to 0x140059FF

The function requires the imageStartAddr input pointer to be 32-bit word aligned. The status is returned by two ways - via a function return as well as by a write to the *isSignVerified pointer. This is provided for redundant protection, the caller shall verify both return values and consider authentic image only when the function returns kStatus_SKBOOT_Success AND *isSignVerified == kSECURE_TRACKER_VERIFIED.

On function output, it returns:

kStatus_SKBOOT_Success when signature verification pass

kStatus_SKBOOT_Fail when parsing certificate header, certificate/certificates chain, RKH or signature verification fails

kStatus_SKBOOT_InvalidArgument for unexpected value in the image

On function output, it writes

*isSignVerified = kSECURE_FALSE (0x5aa55aa5U) when signature verification fails

*isSignVerified = kSECURE_TRACKER_VERIFIED (0x55aacc33U) when signature verification pass

7.4.2 HASH_IRQHandler

This function shall be called from user Hash-crypt interrupt handler, if the sktoot_authenticate() or kb_process() ROM function is called in user applications. The hashing in ROM is implemented as non-blocking and so the interrupt handler function is required to assist with fetching data for Hash-crypt hardware module.

7.5 PRINCE ROM API

The boot ROM API supports PRINCE encryption regions configuration. This section describes all PRINCE-related functions that can be called from the user application.

The whole ROM API table locates at address 0x130010f0 and the PRINCE ROM API part locates at address 0x1300119C.

The bus crypto engine (PRINCE) ROM API prototypes are:

typedef struct BusCryptoEngineInterface

{

skboot_status_t (*bus_crypto_engine_gen_new_iv)(uint32_t region, uint8_t *iv_code, secure_bool_t store, flash_config_t *flash_context);

skboot_status_t (*bus_crypto_engine_load_iv)(uint32_t region, uint8_t *iv_code);

skboot_status_t (*bus_crypto_engine_set_encrypt_for_address_range)(uint8_t region_number, uint32_t start_address, uint32_t length, flash_config_t *flash_context);

} bus_crypto_engine_interface_t;

The skboot_status_t is defined here:

```
typedef enum _skboot_status
{
    kStatus_SKBOOT_Success = 0x5ac3c35au,
    kStatus_SKBOOT_Fail = 0xc35ac35au,
    kStatus_SKBOOT_InvalidArgument = 0xc35a5ac3u,
    kStatus_SKBOOT_KeyStoreMarkerInvalid = 0xc3c35a5au,
} skboot_status_t;
```

The skboot_bool_t is defined here:

```
typedef enum _secure_bool
{
    kSECURE_TRUE = 0xc33cc33cU,
    kSECURE_FALSE = 0x5aa55aa5U,
} secure_bool_t;
```

7.5.1 bus_crypto_engine_gen_new_iv

This API is used for generating new IV (initial vector) code and storing it into the persistent memory. The flash_init ROM API function must be called before calling this PRINCE API.

Prototype

```
skboot_status_t (*bus_crypto_engine_gen_new_iv)(uint32_t region, uint8_t *iv_code, secure_bool_t store, flash_config_t *flash_context);
```

Table 192. Parameters

Parameter	Description
region	Bus encryption engine region index (0, 1, 2).
iv_code	IV code pointer used for storing the newly generated IV code.
store	Flag to allow storing the newly generated IV code into the persistent memory (PFR). Can be assigned to kSECURE_TRUE or kSECURE_FALSE.
flash_context	Pointer to the flash driver context structure initialized by flash_init ROM API function.

Example:

```
#define ROM_API_TREE ((*uint32_t)0x130010f0)
#define FLASH_API_TREE (flash_driver_interface_t*) ROM_API_TREE[3]
#define PRINCE_API_TREE (bus_crypto_engine_interface_t*) ROM_API_TREE[9]
status_t status;
skboot_status_t skboot_status;
```

```
flash_config_t flashConfig;

uint8_t prince_iv_code[52] = {0};

status = FLASH_API_TREE->flash_init(&flashConfig);

skboot_status = PRINCE_API_TREE->bus_crypto_engine_gen_new_iv (0,
&prince_iv_code[0], kSECURE_TRUE, &flashConfig);
```

7.5.2 bus_crypto_engine_load_iv

This API function enables IV code loading into a bus encryption engine (PRINCE) registers.

Prototype

```
skboot_status_t (*bus_crypto_engine_load_iv)(uint32_t region, uint8_t *iv_code);
```

Table 193. Parameters

Parameter	Description
region	Bus encryption engine region index (0, 1, 2).
iv_code	IV code pointer used for passing the IV code.

Example:

```
#define ROM_API_TREE ((*uint32_t)0x130010f0)

#define PRINCE_API_TREE (bus_crypto_engine_interface_t*) ROM_API_TREE[9]

skboot_status_t skboot_status;

uint8_t prince_iv_code[52];

skboot_status = PRINCE_API_TREE->bus_crypto_engine_load_iv(0,
&prince_iv_code[0]);
```

bus_crypto_engine_set_for_address_range: This API function allows the encryption/decryption for specified address range. It configures the PRINCE registers and related PFR regions. The flash_init ROM API function must be called before calling this PRINCE API. Note that the PRINCE configuration can be also done using the blhost ISP command interface, see Section 7.5.4.2.2 “PRINCE region configuration with blhost”.

```
skboot_status_t (*bus_crypto_engine_set_encrypt_for_address_range)(uint8_t
region_number, uint32_t start_address, uint32_t length, flash_config_t *flash_context);
```

Table 194. Parameters

Parameter	Description
region_number	Bus encryption engine region index (0, 1, 2).
start_address	Start address of the area to be encrypted/decrypted
length	Length of the area to be encrypted/decrypted
flash_context	Pointer to the flash driver context structure initialized by flash_init ROM API function.

Example:

```
#define ROM_API_TREE ((*uint32_t)0x130010f0)

#define FLASH_API_TREE (flash_driver_interface_t*) ROM_API_TREE[3]

#define PRINCE_API_TREE (bus_crypto_engine_interface_t*) ROM_API_TREE[9]

status_t status;

skboot_status_t skboot_status;

flash_config_t flashConfig;

status = FLASH_API_TREE->flash_init(&flashConfig);

skboot_status =
PRINCE_API_TREE->bus_crypto_engine_set_encrypt_for_address_range(0, 0, 0x2000,
&flashConfig);
```

For 1.0 silicone version:

```
typedef skboot_status_t
(*BUS_CRYPTO_ENGINE_SET_ENCRYPT_FOR_ADDRESS_RANGE_FPTR)(uint8_t,
uint32_t, uint32_t);

typedef struct BusCryptoEngineInterfaceExt

{

   BUS_CRYPTO_ENGINE_SET_ENCRYPT_FOR_ADDRESS_RANGE_FPTR
bus_crypto_engine_set_encrypt_for_address_range;

} bus_crypto_engine_interface_ext_t;

const bus_crypto_engine_interface_ext_t s_busCryptoEngineInterfaceExt = {

   .bus_crypto_engine_set_encrypt_for_address_range =
(BUS_CRYPTO_ENGINE_SET_ENCRYPT_FOR_ADDRESS_RANGE_FPTR)0x13003d
fb,

};

skboot_status_t skboot_status;

skboot_status =
s_busCryptoEngineInterfaceExt.bus_crypto_engine_set_encrypt_for_address_range(0,
0, 0x2000);
```

7.5.3 ROM TrustZone support

7.5.3.1 Trustzone image type

From TrustZone perspective the ROM distinguishes between two image types:

- TrustZone disabled image
- TrustZone enabled image

The TrustZone Image type is defined in the vector section of image header at offset 0x24, bit 14 (TZM_IMAGE_TYPE):

Table 195. Trustzone image type

TZM_IMAGE_TYPE value (offset 24, bit 14)	
0	TrustZone enabled image
1	TrustZone disabled image

7.5.3.1.1 TrustZone disabled image

TrustZone disabled image is an image, which is supposed to be executed on devices without TrustZone (M33 without security extension). To keep full software compatibility between CM33 with and without security extension, this software/image must be executed in normal mode. To allow easy transition between devices with and without security extension, the LPC55S6x/LPC55S2x/LPC552x ROM supports direct execution of software developed for MC33 devices without security extension. If the TrustZone disabled image is executed, the ROM, before it jumps to user application, configures all device resources into normal world, lock access to all TrustZone related configuration registers, switches from secure to normal world and finally jumps to user application. This mode allows easy reuse of the software developed for Cortex-M33 without security extension. The user doesn't need to do any software modification.

Note: After jump into user application, the security extension (TrustZone) is still enabled. The MCU is running in normal, mode, all TrustZone related configuration registers are locked, memory region 0x13000000-0x13001000 (first 4kB of ROM) is configured as secure memory. The user should avoid code execution or data read from this memory otherwise HardFault will be generated.

7.5.3.1.2 TrustZone enabled image

TrustZone enabled image is an image, which is supposed to be executed on devices with TrustZone (M33 with security extension) and it utilizes TrustZone. It this case the user application is split into secure and non-secure part, and after device reset, the software execution starts in secure mode. If TrustZone enabled image is executed, the ROM doesn't provide any TrustZone settings (except optional TrustZone preset data configuration) and jumps into user application in secure mode. The executed software/image is responsible for TrustZone settings and jump from secure to normal world.

7.5.3.2 TrustZone preset data

LPC55S6x/LPC55S2x/LPC552x ROM provides support for TrustZone data configuration during boot process. The TrustZone preset data includes:

- VTOR, VTOR_NS, NVIC_ITNS0, NVIC_ITNS1 (CPU0) registers
- VTOR (CPU1) register
- Secure MPU
- Non-secure MPU
- SAU
- Secure AHB Controller

If the TrustZone preset is enabled, the ROM, after image validation, configures all TrustZone related registers by data, provided at the end of the image. If any register or whole peripheral has lock feature and corresponding bit is set, the register is also locked, so any further register modification is not possible until next reset.

This feature increases robustness of the user application since the user application jumps into pre-configured TrustZone environment and it doesn't need to contain any TrustZone configuration code.

7.5.3.2.1 TrustZone preset data structure

The TrustZone preset data structure is defined by following C structure:

```
typedef struct _tzm_secure_config
{
    uint32_t cm33_vtor_addr;          /*!< CM33 Secure vector table address */
    uint32_t cm33_vtor_ns_addr;       /*!< CM33 Non-secure vector table address */
    uint32_t cm33_nvic_itns0;         /*!< CM33 Interrupt target non-secure register 0 */
    uint32_t cm33_nvic_itns1;         /*!< CM33 Interrupt target non-secure register 1 */
    uint32_t mcm33_vtor_addr;         /*!< MCM33 Secure vector table address */
    uint32_t cm33_mpu_ctrl;           /*!< MPU Control Register.*/
    uint32_t cm33_mpu_mair0;          /*!< MPU Memory Attribute Indirection Register 0*/
    uint32_t cm33_mpu_mair1;          /*!< MPU Memory Attribute Indirection Register 1
*/
    uint32_t cm33_mpu_rbar0;          /*!< MPU Region 0 Base Address Register */
    uint32_t cm33_mpu_rlar0;          /*!< MPU Region 0 Limit Address Register */
    uint32_t cm33_mpu_rbar1;          /*!< MPU Region 1 Base Address Register */
    uint32_t cm33_mpu_rlar1;          /*!< MPU Region 1 Limit Address Register */
    uint32_t cm33_mpu_rbar2;          /*!< MPU Region 2 Base Address Register */
    uint32_t cm33_mpu_rlar2;          /*!< MPU Region 2 Limit Address Register */
    uint32_t cm33_mpu_rbar3;          /*!< MPU Region 3 Base Address Register */
    uint32_t cm33_mpu_rlar3;          /*!< MPU Region 3 Limit Address Register */
    uint32_t cm33_mpu_rbar4;          /*!< MPU Region 4 Base Address Register */
    uint32_t cm33_mpu_rlar4;          /*!< MPU Region 4 Limit Address Register */
    uint32_t cm33_mpu_rbar5;          /*!< MPU Region 5 Base Address Register */
    uint32_t cm33_mpu_rlar5;          /*!< MPU Region 5 Limit Address Register */
    uint32_t cm33_mpu_rbar6;          /*!< MPU Region 6 Base Address Register */
```

```
    uint32_t cm33_mpu_rlar6;            /*!< MPU Region 6 Limit Address Register */
    uint32_t cm33_mpu_rbar7;            /*!< MPU Region 7 Base Address Register */
    uint32_t cm33_mpu_rlar7;            /*!< MPU Region 7 Limit Address Register */
    uint32_t cm33_mpu_ctrl_ns;          /*!< Non-secure MPU Control Register.*/
    uint32_t cm33_mpu_mair0_ns;         /*!< Non-secure MPU Memory Attribute
Indirection Register 0 */
    uint32_t cm33_mpu_mair1_ns;         /*!< Non-secure MPU Memory Attribute
Indirection Register 1 */
    uint32_t cm33_mpu_rbar0_ns;         /*!< Non-secure MPU Region 0 Base Address
Register */
    uint32_t cm33_mpu_rlar0_ns;         /*!< Non-secure MPU Region 0 Limit Address
Register */
    uint32_t cm33_mpu_rbar1_ns;         /*!< Non-secure MPU Region 1 Base Address
Register */
    uint32_t cm33_mpu_rlar1_ns;         /*!< Non-secure MPU Region 1 Limit Address
Register */
    uint32_t cm33_mpu_rbar2_ns;         /*!< Non-secure MPU Region 2 Base Address
Register */
    uint32_t cm33_mpu_rlar2_ns;         /*!< Non-secure MPU Region 2 Limit Address
Register */
    uint32_t cm33_mpu_rbar3_ns;         /*!< Non-secure MPU Region 3 Base Address
Register */
    uint32_t cm33_mpu_rlar3_ns;         /*!< Non-secure MPU Region 3 Limit Address
Register */
    uint32_t cm33_mpu_rbar4_ns;         /*!< Non-secure MPU Region 4 Base Address
Register */
    uint32_t cm33_mpu_rlar4_ns;         /*!< Non-secure MPU Region 4 Limit Address
Register */
    uint32_t cm33_mpu_rbar5_ns;         /*!< Non-secure MPU Region 5 Base Address
Register */
    uint32_t cm33_mpu_rlar5_ns;         /*!< Non-secure MPU Region 5 Limit Address
Register */
    uint32_t cm33_mpu_rbar6_ns;         /*!< Non-secure MPU Region 6 Base Address
Register */
    uint32_t cm33_mpu_rlar6_ns;         /*!< Non-secure MPU Region 6 Limit Address
Register */
```

```
    uint32_t cm33_mpu_rbar7_ns;          /*!< Non-secure MPU Region 7 Base Address
Register */
    uint32_t cm33_mpu_rlar7_ns;          /*!< Non-secure MPU Region 7 Limit Address
Register */
uint32_t cm33_sau_ctrl;              /*!< SAU Control Register.*/
    uint32_t cm33_sau_rbar0;             /*!< SAU Region 0 Base Address Register */
    uint32_t cm33_sau_rlar0;             /*!< SAU Region 0 Limit Address Register */
    uint32_t cm33_sau_rbar1;             /*!< SAU Region 1 Base Address Register */
    uint32_t cm33_sau_rlar1;             /*!< SAU Region 1 Limit Address Register */
    uint32_t cm33_sau_rbar2;             /*!< SAU Region 2 Base Address Register */
    uint32_t cm33_sau_rlar2;             /*!< SAU Region 2 Limit Address Register */
    uint32_t cm33_sau_rbar3;             /*!< SAU Region 3 Base Address Register */
    uint32_t cm33_sau_rlar3;             /*!< SAU Region 3 Limit Address Register */
    uint32_t cm33_sau_rbar4;             /*!< SAU Region 4 Base Address Register */
    uint32_t cm33_sau_rlar4;             /*!< SAU Region 4 Limit Address Register */
    uint32_t cm33_sau_rbar5;             /*!< SAU Region 5 Base Address Register */
    uint32_t cm33_sau_rlar5;             /*!< SAU Region 5 Limit Address Register */
    uint32_t cm33_sau_rbar6;             /*!< SAU Region 6 Base Address Register */
    uint32_t cm33_sau_rlar6;             /*!< SAU Region 6 Limit Address Register */
    uint32_t cm33_sau_rbar7;             /*!< SAU Region 7 Base Address Register */
    uint32_t cm33_sau_rlar7;             /*!< SAU Region 7 Limit Address Register */
    uint32_t flash_rom_slave_rule;       /*!< FLASH/ROM Slave Rule Register 0 */
    uint32_t flash_mem_rule0;            /*!< FLASH Memory Rule Register 0 */
    uint32_t flash_mem_rule1;            /*!< FLASH Memory Rule Register 1 */
    uint32_t flash_mem_rule2;            /*!< FLASH Memory Rule Register 2 */
    uint32_t rom_mem_rule0;              /*!< ROM Memory Rule Register 0 */
    uint32_t rom_mem_rule1;              /*!< ROM Memory Rule Register 1 */
    uint32_t rom_mem_rule2;              /*!< ROM Memory Rule Register 2 */
    uint32_t rom_mem_rule3;              /*!< ROM Memory Rule Register 3 */
    uint32_t ramx_slave_rule;            /*!< RAMX Slave Rule Register */
    uint32_t ramx_mem_rule0;             /*!< RAMX Memory Rule Register 0 */
```

```
    uint32_t ram0_slave_rule;          /*!< RAM0 Slave Rule Register */
    uint32_t ram0_mem_rule0;           /*!< RAM0 Memory Rule Register 0 */
    uint32_t ram0_mem_rule1;           /*!< RAM0 Memory Rule Register 1 */

    uint32_t ram1_slave_rule;          /*!< RAM1 Slave Rule Register */
    uint32_t ram1_mem_rule0;           /*!< RAM1 Memory Rule Register 0 */
    uint32_t ram1_mem_rule1;           /*!< RAM1 Memory Rule Register 1 */
    uint32_t ram2_slave_rule;          /*!< RAM2 Slave Rule Register */
    uint32_t ram2_mem_rule0;           /*!< RAM2 Memory Rule Register 0 */
    uint32_t ram2_mem_rule1;           /*!< RAM2 Memory Rule Register 1 */
    uint32_t ram3_slave_rule;          /*!< RAM3 Slave Rule Register */
    uint32_t ram3_mem_rule0;           /*!< RAM3 Memory Rule Register 0 */
    uint32_t ram3_mem_rule1;           /*!< RAM3 Memory Rule Register 1 */
    uint32_t ram4_slave_rule;          /*!< RAM4 Slave Rule Register */
    uint32_t ram4_mem_rule0;           /*!< RAM4 Memory Rule Register 0 */
    uint32_t apb_grp_slave_rule;       /*!< APB Bridge Group Slave Rule Register */
   uint32_t apb_grp0_mem_rule0;        /*!< APB Bridge Group 0 Memory Rule Register 0
*/
   uint32_t apb_grp0_mem_rule1;        /*!< APB Bridge Group 0 Memory Rule Register 1
*/
   uint32_t apb_grp0_mem_rule2;        /*!< APB Bridge Group 0 Memory Rule Register 2
*/
   uint32_t apb_grp0_mem_rule3;        /*!< APB Bridge Group 0 Memory Rule Register 3
*/
   uint32_t apb_grp1_mem_rule0;        /*!< APB Bridge Group 1 Memory Rule Register 0
*/
   uint32_t apb_grp1_mem_rule1;        /*!< APB Bridge Group 1 Memory Rule Register 1
*/
   uint32_t apb_grp1_mem_rule2;        /*!< APB Bridge Group 1 Memory Rule Register 2
*/
   uint32_t apb_grp1_mem_rule3;        /*!< APB Bridge Group 1 Memory Rule Register 3
*/
```

```
    uint32_t ahb_periph0_slave_rule0;     /*!< AHB Peripherals 0 Slave Rule Register 0 */
    uint32_t ahb_periph0_slave_rule1;     /*!< AHB Peripherals 0 Slave Rule Register 1 */
    uint32_t ahb_periph1_slave_rule0;     /*!< AHB Peripherals 1 Slave Rule Register 0 */
    uint32_t ahb_periph1_slave_rule1;     /*!< AHB Peripherals 1 Slave Rule Register 1 */
    uint32_t ahb_periph2_slave_rule0;     /*!< AHB Peripherals 2 Slave Rule Register 0 */
    uint32_t ahb_periph2_slave_rule1;     /*!< AHB Peripherals 2 Slave Rule Register 1 */
   uint32_t ahb_periph2_mem_rule0;      /*!< AHB Peripherals 2 Memory Rule Register 0*/
  uint32_t usb_hs_slave_rule0;         /*!< HS USB Slave Rule Register 0 */
    uint32_t usb_hs__mem_rule0;          /*!< HS USB Memory Rule Register 0 */
    uint32_t sec_gp_reg0;               /*!< Secure GPIO Register 0 */
    uint32_t sec_gp_reg1;               /*!< Secure GPIO Register 1 */
    uint32_t sec_gp_reg2;               /*!< Secure GPIO Register 2 */
    uint32_t sec_gp_reg3;               /*!< Secure GPIO Register 3 */
    uint32_t sec_int_reg0;              /*!< Secure Interrupt Mask for CPU1 Register 0 */
    uint32_t sec_int_reg1;              /*!< Secure Interrupt Mask for CPU1 Register 1 */
    uint32_t sec_gp_reg_lock;           /*!< Secure GPIO Lock Register */
    uint32_t master_sec_reg;            /*!< Master Secure Level Register */
    uint32_t master_sec_anti_pol_reg;     /*!< Master Secure Level Anti-pole Register */
    uint32_t cm33_lock_reg;             /*!< CM33 Lock Control Register */
    uint32_t mcm33_lock_reg;              /*!< MCM33 Lock Control Register */
     uint32_t misc_ctrl_dp_reg;          /*!< Secure Control Duplicate Register */
    uint32_t misc_ctrl_reg;             /*!< Secure Control Register */
    uint32_t misc_tzm_settings;          /*!< Miscellaneous TZM settings */
} tzm_secure_config_t;
```

The configuration data are copied one to one into appropriate registers except misc_tzm_settings. The configuration word misc_tzm_settings is defined in following table

Table 196. Misc TZM settings

Field	Function
31-1	Reserved
0 SECUREFAULTENA	SHCSR.SECUREFAULTENA control 0b - SECUREFAULTENA is set to 0 1b - SECUREFAULTENA is set to 1

The configuration data are attached in binary format at the end of the image, in the case of signed image in front of signature, see figure xx.

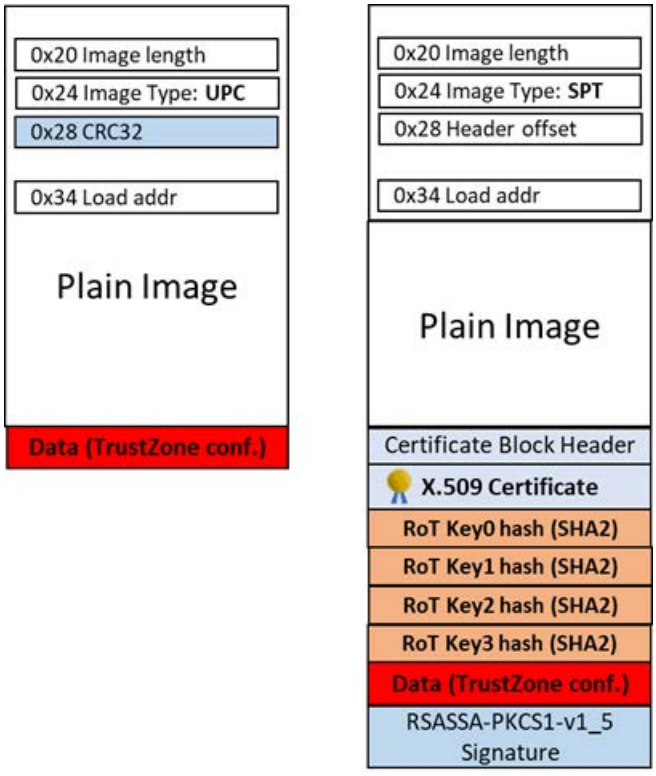


Fig 19. Location of TrustZone configuration data in the image file

The user can also use elftosb.exe tool to create TrustZone configuration data. For more information please see elftosb manual.

The information whether image file contains TrustZone configuration data or not is defined in the vector section of the image header at offset 0x24, bit 13 (TZM_PRESET)

Table 197. TZM Preset value

TZM_PRESET value (offset 24, bit 13)	
0	TrustZone data not present
1	TrustZone data present

Note: Since the TrustZone configuration is enabled before a jump to the user application, the user's TrustZone configuration data must allow ROM code execution for successful transition from secure to normal mode and jump to user application. This means that user's TrustZone settings must include:

1. Whole ROM space (0x13000000-0x1301FFFF) must be configured as secure privilege.

2. In the case, that secure MPU is used, whole ROM space (0x13000000-0x1301FFFF) must be configured for code execution.

If these two conditions are not met, boot process will fail.

7.5.3.2.2 TrustZone image type restriction control during boot process

The user can restrict, which TrustZone image type is allowed for execution. For this purpose there are two bits CMPA.SECURE_BOOT_CFG.TZM_IMAGE_TYPE[1:0] in Customer Manufacturing Programmed Area (CMPA) flash. This field restricts execution of the TrustZone image type as described in following table:

Table 198. Allowed Trustzone image types

Allowed TrustZone Image Type	CMPA.SECURE_BOOT_CFG.TZM_IMAGE_TYPE[1:0] Value
Any TrustZone image type is allowed. The image type is set in the Vectors section of the image (offset 0x24)	00b
TrustZone disabled image type is allowed only	01b
TrustZone enabled or TrustZone enabled with TrustZone Preset Data image type are allowed	10b
TrustZone enabled with Preset Data image type is allowed	11b

7.5.3.3 Boot ROM API and TrustZone

7.5.3.3.1 TrustZone disabled images

TrustZone disabled image is executed in normal mode and whole memory space is configured as non-secure (except first 4kB of ROM). Thus, the ROM API can be used without any limitation as on any LPC device without security extension.

7.5.3.3.2 TrustZone enabled images

The whole boot ROM is executed in secure mode. Thus, the user can fully control which ROM API will be available also in normal world. The user can expose full ROM API, limited ROM API functions set or modify/limit ROM API functionality. For example, the user can expose flash programming API with limitation to non-secure data memory address range only. It means that code executed in normal world can program data into flash memory, but it cannot erase whole flash or reprogram application itself or its secure part.

To enable the ROM API into normal world the user must create entry function for every ROM API function exposed to normal world. Example of entry function for flash programming can be seen below:

```
#define ROM_API_TREE ((*uint32_t)0x130010f0)

#define FLASH_API_TREE ((flash_driver_interface_t*) ROM_API_TREE[3])

__cmse_nonsecure_entry status_t flash_program_NSE(flash_config_t *config, uint32_t start, uint8_t *src, uint32_t lengthInBytes)
{

 status_t status;

 /*

 Validate all input parameters based on application requirements. If input parameters are
```

```
  Invalid, return error
  */
  status = FLASH_API_TREE->flash_program(&config, start, src, lengthInBytes);
  return status;
}
```

Then user can call flash program function from normal world as:

```
flash_config_t flashConfig;
uint8_t programBuffer[512];
status = flash_program_NSE(&flashConfig, 0x0, programBuffer, sizeof(programBuffer));
```

7.5.4 Secure boot usage

The LPC55S6x/LPC55S2x/LPC552x allows booting of public-key signed images. The device boot ROM supports following types of security protected boot modes:

- Secure boot with signed image
- Secure boot with signed image from encrypted internal flash regions

Each of these options has attributes related to manufacturability, the firmware update scheme and level of protection against attacks.

The ROM further supports public keys and image revocation i.e. the method of not allowing new updates to be applied unless they are of a specific version. This is the basis for roll back protection.

The following section describes the main steps for key provisioning, creating signed images and loading the signed images into the target. Tools used are elftosb or elftosb-gui – see AN12283 LPC55S6x/LPC55S2x/LPC552x Secure Boot for detailed step-by-step guide describing use of these tools.

7.5.4.1 Keys and certificates

Image signing process will require RSA key pair and image signing certificate. Use e.g. openssl for key and certificate generation.

ROM supports:

- Up to 4 Root of Trust (RoT) keys
- Up to 16 Image key certificates with Image revocation feature

Prior the secure image preparation Root Key Hash table needs to be written to corresponding CFPA boot pages

7.5.4.1.1 CFPA/CMPA page preparation

Before the first use of the device CFPA and CMPA pages are cleared, there are registers related to secure boot which must be set up.

ROTKH_REVOKE field at CFPA page address 0x9DE18 must be setup to accept signed images with created certificates

- Enter ISP boot mode by asserting ISP boot pin
- Prepare CMPA page using elftosb-gui PC tool
  - RKTH field containing root key table hash
  - SEC_BOOT_EN secure boot enable bit
  - RSA4K field selecting minimal key length
- Prepare CFPA page using elftosb-gui PC tool
  - RKTH_REVOKE field to accept signed images with created certificates
- Write prepared CFPA/CMPA page into flash memory using blhost tool

7.5.4.1.2 Signed image preparation

NXP provides elftosb and elftosb-gui tools which prepare signed binary, which can be loaded to target device. The input for elftosb program are plain application image in binary format, image signing certificate, associated private key and JSON format configuration structure. For detailed step-by-step guide see AN12283 LPC55S6x/LPC55S2x/LPC552x Secure Boot application note.

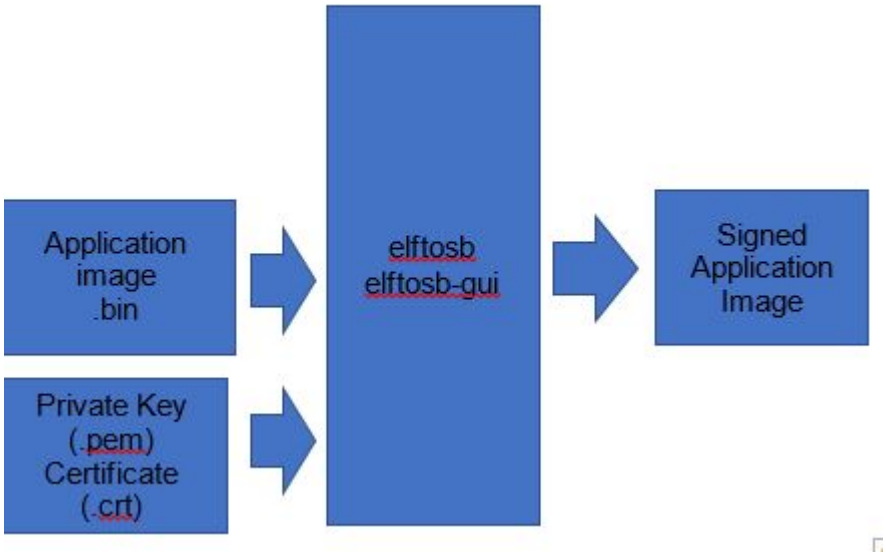


Fig 20. Signed Image Preparation

Following information are needed by elftosb tool to produce Internal flash (XIP) signed image:

- Plain application binary generated for LPC55S6x/LPC55S2x/LPC552x device
- Start address of application binary
- TZ related settings
- Certificates or chain certificates

- Private key for selected certificate (last certificate in chain)

7.5.4.1.3 Loading signed image

The signed image could be programmed directly into the device using various methods. The main options are:

- ROM In System Programming (ISP) using write-memory blhost command
- ROM ISP using Secure FW update container
- Programming signed image directly from target application using ROM API
- Flashing signed image through debugger

7.5.4.2 Internal flash encryption using PRINCE engine

Boot ROM offers configuration PRINCE engine during the boot time.

First, during the flash programming, PRINCE engine is set up to store the image in encrypted format using user defined key.

The user key is stored on device and protected against copying using PUF encryption in the format readable only for given instance of the processor. SRAM PUF internally uses HW specific random keys unique for each processor.

During the boot ROM will locate the key store, decrypt it internally using SRAM PUF and pre-configure the PRINCE engine. Keys are delivered to the PRINCE engine through internal HW bus. User application is then decrypted in real time and executed.

The following steps describes the tools required to perform key provisioning and programming application image using PRINCE.

7.5.4.2.1 PRINCE related PUF key store setup

The keys used for PRINCE encryption/decryption are generated in the device using on-chip SRAM PUF and they are delivered to the PRINCE engine through internal hardware bus.

In the following example you can see the sequence of commands to be issued from PC blhost application to the device in ISP mode to generate proper PRINCE enabled Key Store. The key store is saved into device PFR and accessed by boot ROM during secure boot.

In this example blhost PC tool is connecting to the processor using UART COM6 and baudrate 38400.

- generate device activation code and store it into key store structure

blhost -p com6,38400 -- key-provisioning enroll

- generate random PRINCE region 0. PRINCE region 0 key type = 7

blhost -p com6,38400 -- key-provisioning set_key 7 16

- generate random PRINCE region 1. PRINCE region 1 key type = 8

blhost -p com6,38400 -- key-provisioning set_key 8 16

- generate random PRINCE region 2. PRINCE region 0 key type = 9

blhost -p com6,38400 -- key-provisioning set_key 9 16

- save the key store into PFR page of Flash memory

blhost -p com6,38400 -- key-provisioning write_key_nonvolatile 0

7.5.4.2.2 PRINCE region configuration with blhost

For PRINCE encryption and decryption, the regions and sub-regions for the crypto operation needs to be configured. This can be done with ISP command “configure-memory”. This command has to be called with following data structure.

Table 199. Structure for configure-memory command

Offset	Size	Description
0	4	PRINCE Configuration
4	8	PRINCE Region info

Table 200. PRINCE configuration register for configure-memory command

Bit	Symbol
1:0	0x00 – PRINCE Region 0 0x01 – PRINCE Region 1 0x10 – PRINCE Region 2
25:2	Reserved
31:8	0x50 ('P') – Configure PRINCE

Table 201. PRINCE region info register for configure-memory command

Bit	Symbol
31:0	PRINCE Region X Start
63:32	PRINCE Region X size

Load structure into RAM memory and call “configure-memory” command with this structure:

1. Region selection (Region 0 in this example
   - blhost.exe -p COMxx -- fill-memory 0x20034000 4 0x50000000
2. Start address of encrypted area (Address 0x0 in this example)
   - blhost.exe -p COMxx -- fill-memory 0x20034004 4 0
3. Length of the encrypted area (0x40000 in this example)
   - blhost.exe -p COMxx -- fill-memory 0x20034008 4 0x40000
4. Call configure-memory with prepared structure in RAM
   - blhost.exe -p COMxx -- configure-memory 0 0x20034000

Following this command, PRINCE is configured for flash encryption.

Note: The PFR area should be excluded from PRINCE encryption area, i.e., the start and size settings in the configuration for the structure must be set to avoid overlapping with the PFR area.

7.5.4.2.3 Upload image

A "prince erase checker" is implemented in the boot ROM that checks whether the whole PRINCE sub region (8k block) are erased at once. Similarly, "prince flash write checker" is implemented in the ROM code to check whether the whole PRINCE 8 KB subregions are programmed at once. To load the image that is on-the-fly encrypted by PRINCE, the following sequence of ISP commands need to be issued using BLHOST tool:

1. Erase the flash memory
   - blhost.exe -p COMxx -- flash-erase-region 0x00000 0x40000
2. Load the image into the flash
   - blhost.exe -p COMxx -- write-memory 0 <path to the image(.bin)>
3. Reset device
   - Press reset pin or run BLHOST tool blhost -p COMxx reset
4. The image loaded into flash starts executing

Note: Special care should be taken when erasing and writing the encrypted flash area. The whole Prince encrypted flash area must be erased and written in a single operation.

Note: Because the "prince erase checker" and “prince flash write” are implemented in the boot ROM, the flash-erase-region command and write-memory command has to cover the whole encrypted area previously defined through the configure-memory command. In other words, the image (.bin) size has to equal to the encrypted area size (8k aligned).

After these steps the image loaded in the flash is encrypted. Decryption of the flash content is done on-the-fly by the PRINCE hardware engine.

Note: Due to several limitations in the "prince erase checker" and "prince flash write checker" implementations in the boot ROM, it is recommend to properly check the flash-erase-region and write-memory command parameters to strictly cover the whole encrypted area. Using the latest MCUXpressoSDK Prince driver is the preferred way for performing Prince region configuration, erase, and write operations.

8.1 How to read this chapter

All LPC55S6x/LPC55S2x/LPC552x devices include In-System Programming (ISP) functions to support image programming from serial interface (UART, I^2C, SPI) and USB HID. In-Application Programming (IAP) calls are available.

8.2 Features

- In-System Programming supports:
  - Supports UART, I^2C, SPI, and USB peripheral interfaces.
  - Automatic detection of the active peripheral.
  - UART peripheral implements auto-baud.
  - Common packet-based protocol for all peripherals.
  - Packet error detection and retransmit.
  - Flash-resident configuration options.
  - Protection of RAM used by the bootloader, while it is running.
  - Provides a command to read the properties of the device, such as Flash and RAM size.
  - Multiple options for executing the bootloader either at system startup or under application control at runtime.
  - Support for internal flash.
  - Support for encrypted image download.

8.3 General description

8.3.1 Bootloader

The internal ROM memory is used to store the boot code. After a reset, the ARM processor starts its code execution from this memory.The bootloader code is executed every time the part is powered-on, is reset, or is woken up from a deep power-down, low power mode.

The bootloader provides flash programming utility that operates over a serial connection on the MCUs. It enables quick and easy programming of MCUs through the entire product lifecycle, including application development, final product manufacturing, and beyond. Host-side command line and GUI tools are available to communicate with the bootloader. Users can utilize host tools to upload/download application code and do manufacturing via the bootloader.

For the bootloader operation and boot pin, see Chapter 6 “LPC55S6x/LPC55S2x/LPC552x Boot ROM”

Remark: Flash operations (erase, blank check, program) and reading a single word can only be performed for CPU frequencies of up to 100 MHz. These operations cannot be performed for frequencies above 100 MHz.

8.3.2 In-System Programming (ISP) and In-Application Programming (IAP)

Serial booting and other related functions, are supported in several different ways:

- For details of the ISP protocol, see Section 8.4 “In-System programming protocol”.
- For details of the ISP packet, see Section 8.5 “Bootloader packet types”.
- For details of the ISP commands, seeSection 8.6 “The bootloader command set”.
- For details of UART In-System Programming, see Section 8.8 “UART ISP”.
- For details of I^2C In-System Programming, see Section 8.9 “I2C In-System Programming”
- For details of SPI In-System Programming, seeSection 8.10 “SPI In-System programming”.
- For details of USB In-System programming, see Section 8.11 “USB In-System Programming”
- For details of In-Application Programming, see Section 8.12 “In-Application-Programming”.

8.3.3 Memory map after any reset

The boot ROM is located in the memory region starting from the address 0x1300 0000. Both the ISP and IAP software use parts of the on-chip RAM. The RAM usage is described in Section 8.3.4 “ISP interrupt and SRAM use”. For more information, see Chapter 6 “LPC55S6x/LPC55S2x/LPC552x Boot ROM”

Based on the DEFAULT_ISP_MODE bit settings or ISP pin settings, the ROM will enter ISP mode and auto-detect activity on the I^2C / SPI/ USART or USB-HID interface. The auto-detect looks for activity on the USART, I^2C, SPI and USB-HID interfaces and selects the appropriate interface once a properly formed frame is received. If an invalid frame is received, the data is discarded and scanning resumes. USART, I^2C, SPI and USB-HID ISP communications are described in Section 8.4 “In-System programming protocol” and Section 8.5 “Bootloader packet types”.

8.3.4 ISP interrupt and SRAM use

8.3.4.1 Interrupts during IAP

When the user application code starts executing, the interrupt vectors from the SRAM are active. Before making any IAP call, disable the interrupts. The IAP code does not use or disable interrupts.

8.3.4.2 RAM used by the ISP command handler

Below regions are reserved for bootloader use when the bootloader is running. The heap and the BSS, RW section need to be reserved for the ROM API use before calling the ROM APIs in user application (IAP scenario).

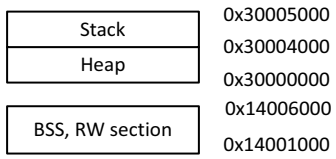


Fig 21. Reserved RAM region for the boot ROM

8.4 In-System programming protocol

This section explains the general protocol for the packet transfers between the host and the bootloader. The description includes the transfer of packets for different transactions, such as commands with no data phase and commands with incoming or outgoing data phase. The next section describes various packet types used in a transaction.

Each command sent from the host is replied to with a response command.

Commands may include an optional data phase.

- If the data phase is incoming (from the host to the bootloader), it is part of the original command.
- If the data phase is outgoing (from the bootloader to host), it is part of the response command.

8.4.1 Command with no data phase

The protocol for a command with no data phase contains:

- Command packet (from the host)
- Generic response command packet (to host)

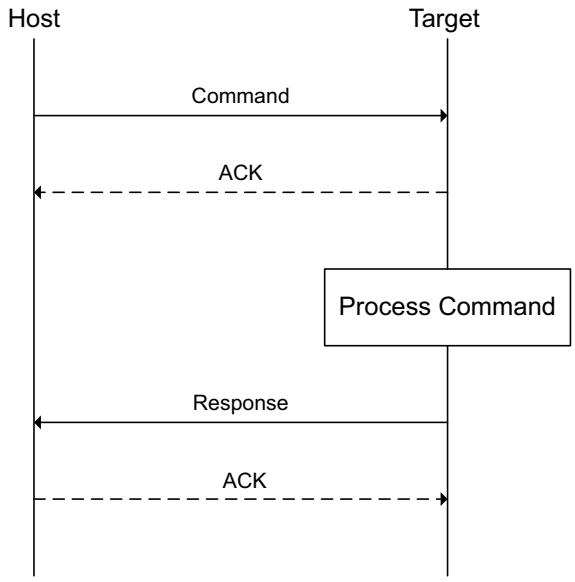


Fig 22. Command with no data phase

Remark: In these diagrams, the ACK sent in response to a command or a data packet can arrive at any time before, during, or after the command or data packet has processed.

8.4.2 Command with the incoming data phase

The protocol for a command with incoming data phase contains:

- Command packet (from host) (kCommandFlag_HasDataPhase set).
- Generic response command packet (to host).
- Incoming data packets (from the host).
- Generic response command packet.

Note:

- The host may not send any further packets while it is waiting for the response to a command.
- The data phase is aborted if the Generic Response packet prior to the start of the Data phase does not have a status of kStatus_Success.
- Data phases may be aborted by the receiving side by sending the final
- GenericResponse early with a status of kStatus_AbortDataPhase. The host may abort the data phase early by sending a zero-length data packet.
- The final Generic Response packet sent after the data phase includes the status of the entire operation.

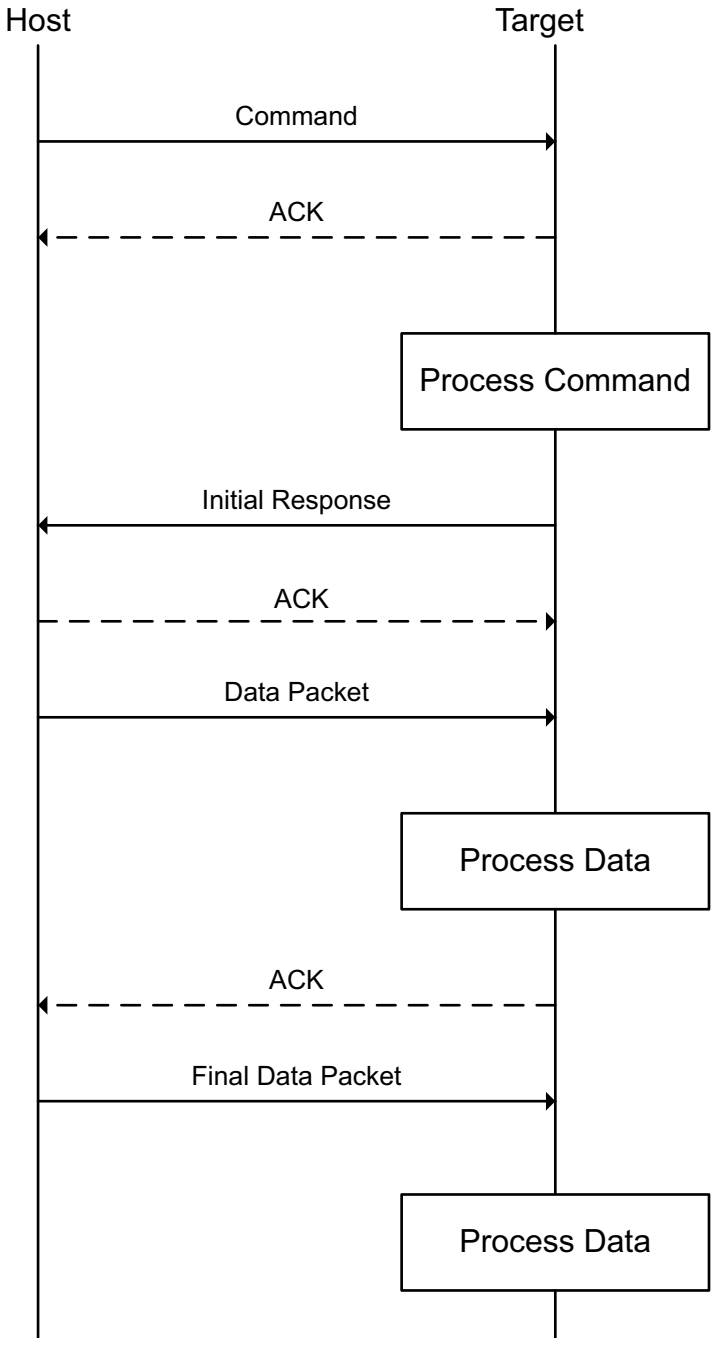


Fig 23. Packet flow command with incoming data phase

8.4.3 Command with outgoing data phase

The protocol for a command with an outgoing data phase contains:

- Command packet (from the host).
- ReadMemory Response command packet (to host) (kCommandFlag_HasDataPhase set).

- Outgoing data packets (to host).
- Generic response command packet (to host).

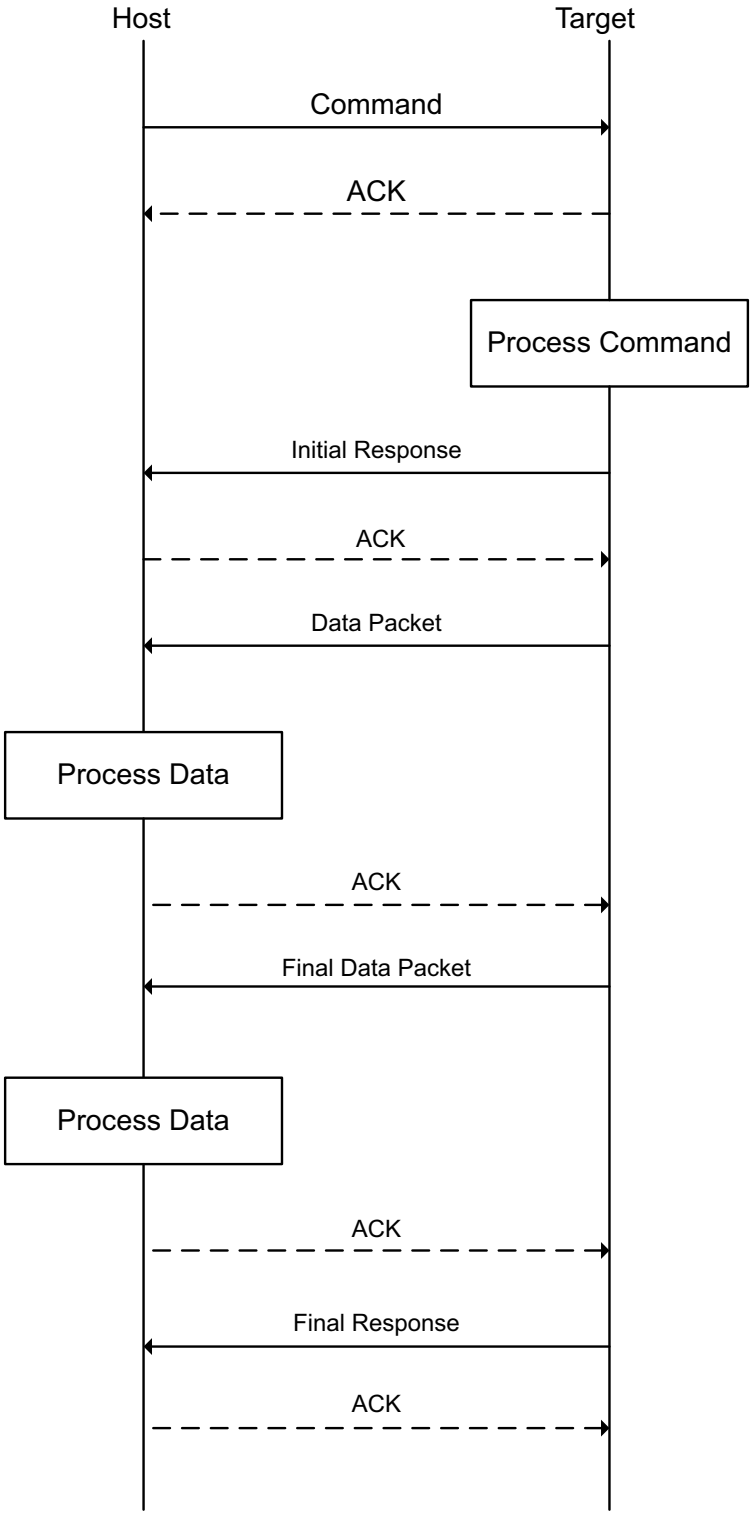


Fig 24. Command with outgoing data phase

Note:

- The data phase is considered part of the response command for the outgoing data phase sequence.
- The host may not send any further packets while the host is waiting for the response to a command.
- The data phase is aborted if the ReadMemory Response command packet, prior to the start of the data phase, does not contain the kCommandFlag_HasDataPhase flag.
- Data phases may be aborted by the host sending the final Generic Response early with a status of kStatus_AbortDataPhase. The sending side may abort the data phase early by sending a zero-length data packet.
- The final Generic Response packet sent after the data phase includes the status of the entire operation.

8.5 Bootloader packet types

8.5.1 Introduction

The bootloader device works in slave mode. All data communications are initiated by a host, which is either a PC or an embedded host. The bootloader device is the target, which receives a command or data packet. All data communications between host and target are packetized.

There are six types of packets used:

- Ping packet.
- Ping Response packet.
- Framing packet.
- Command packet.
- Data packet.
- Response packet.

All fields in the packets are in little-endian byte order.

8.5.2 Ping packet

The Ping packet is the first packet sent from a host to the target to establish a connection on the selected peripheral in order to run autobaud detection. The Ping packet can be sent from host to target at any time that the target is expecting a command packet. If the selected peripheral is UART, a Ping packet must be sent before any other communications. For other serial peripherals, it is optional.

In response to a Ping packet, the target sends a Ping response packet, discussed in the later sections.

Table 202. Ping packet format

Byte #	Value	Name
0	0x5A	Start byte
1	0xA6	Ping

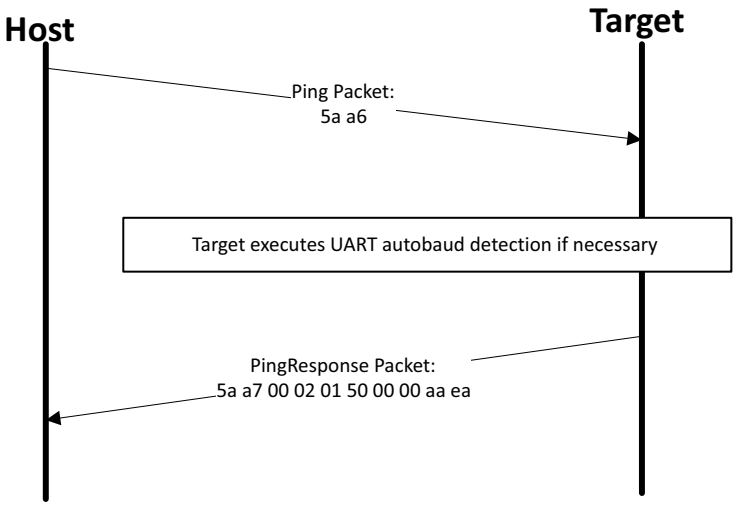


Fig 25. Ping packet protocol sequence

8.5.3 Ping response packet

The target sends a Ping response packet back to the host after receiving a Ping packet. If communication is over a UART peripheral, the target uses the incoming Ping packet to determine the baud rate before replying with the Ping response packet. Once the Ping response packet is received by the host, the connection is established, and the host starts sending commands to the target.

Table 203. Ping response packet format

Byte	Value	Parameter
0	0x5A	Start byte
1	0xA7	Ping response code
2	0x00	Protocol bugfix
3	0x03	Protocol minor
4	0x01	Protocol major
5	0x50	Protocol name = 'P' (0x50)
6	0x00	Options low
7	0x00	Options high
8	0xfb	CRC16 low
9	0x40	CRC16 high

For the UART peripheral, it must be sent by the host when a connection is first established, in order to run outbound. For other serial peripherals, it is optional but recommended to determine the serial protocol version. The version number is in the same format as the bootloader version number returned by the GetProperty command.

8.5.4 Framing packet

The framing packet is used for flow control and error detection for the communications links that do not have such features built-in. The framing packet structure sits between the link layer and the command layer. It wraps command and data packets as well.

Every framing packet containing data sent in one direction results in a synchronizing response framing packet in the opposite direction.

The framing packet described in this section is used for serial peripherals including the UART, I^2C, and SPI. The USB HID peripheral does not use framing packets. Instead, the packetization inherent in the USB protocol itself is used.

Table 204. Framing packet format

Byte	Value	Parameter	Description
0	0x5A	Start byte	
1		PacketType	
2		Length_low	Length is a 16-bit field that specifies the entire command or data packet size in bytes.
3		Length_high	
4		Crc16_low	This is a 16-bit field. The CRC16 value covers entire framing packet, including the start byte and command or data packets, but does not include the CRC bytes. See Section 8.5.5 “CRC16 algorithm”.
5		Crc16_high	
6....n		Command or Data packet payload	

A special framing packet that contains only a start byte and a packet type is used for synchronization between the host and target.

Table 205. Special framing packet format

Byte	Value	Parameter
0	0x5A	Start byte
1	0xA*n*	packetType

The Packet Type field specifies the type of the packet from one of the defined types (below):

Table 206. Packet type field

Packet type	Name	Description
0xA1	kFramingPacketType_Ack	The previous packet was received successfully; the sending of more packets is allowed.
0xA2	kFramingPacketType_Nak	The previous packet was corrupted and must be re-sent.
0xA3	kFramingPacketType_AckAbort	Data phase is being aborted.

Table 206. Packet type field

Packet type	Name	Description
0xA4	kFramingPacketType_Command	The framing packet contains a command packet payload.
0xA5	kFramingPacketType_Data	The framing packet contains a data packet payload.
0xA6	kFramingPacketType_Ping	Sent to verify the other side is alive. Also used for UART autobaud.
0xA7	kFramingPacketType_PingResponse	A response to Ping. It contains the framing protocol version number and options.

8.5.5 CRC16 algorithm

The CRC is computed over each byte in the framing packet header, excluding the CRC16 field itself, and all of the payload bytes. The CRC algorithm is the XMODEM variant of CRC16.

The characteristics of the XMODEM variants are:

Table 207. CRC16 algorithm

Width	16
Polynomial	0x1021
Init value	0x0000
Reflect in	False
Reflect out	False
Xor out	0x0000
Check result	0x31c3

The check result is computed by running the ASCII character sequence "123456789" through the algorithm.

```
uint16_t crc16_update(const uint8_t * src, uint32_t lengthInBytes)
{
    uint32_t crc = 0;
    uint32_t j;
    for (j=0; j < lengthInBytes; ++j)
    {
        uint32_t i;
     uint32_t byte = src[j];
       crc ^= byte << 8;
       for (i = 0; i < 8; ++i)
        {
           uint32_t temp = crc << 1;
           if (crc & 0x8000)
            {
                temp ^= 0x1021;
           }
            crc = temp;
        }
    }
    return crc;
```

}

8.5.6 Command packet

The command packet carries a 32-bit command header and a list of 32-bit parameters.

Table 208. Command packet format

Command Packet Format (32 bytes)										
Command Header (4 bytes)				28 bytes for Parameters (Max 7 parameters)						
Tag	Flags	Rsvd	Param Count	Param 1 (32-bit)	Param 2 (32-bit)	Param 3 (32-bit)	Param 4 (32-bit)	Param 5 (32-bit)	Param 6 (32-bit)	Param 7 (32-bit)

Table 209. Command header format

Byte #	Command header field	Reset value
0	Command or Response tag	The command header is 4 bytes long with these fields.
1	Flags	
2	Reserved. Should be 0x00.	
3	ParameterCount	

The header is followed by 32-bit parameters up to the value of the ParameterCount field specified in the header. Because a command packet is 32 bytes long, only seven parameters can fit into the command packet.

Command packets are also used by the target to send responses back to the host. As mentioned earlier, command packets and data packets are embedded into framing packets for all of the transfers.

Table 210. Command tags

Command tag	Name	Description
0x01	FlashEraseAll	The command tag specifies one of the commands supported by the bootloader. The valid command tags for the bootloader are listed here.
0x02	FlashEraseRegion	
0x03	ReadMemory	
0x04	WriteMemory	
0x05	FillMemory	
0x06	Reserved	
0x07	GetProperty	
0x08	ReceiveSbFile	
0x09	Execute	
0x0A	Call	
0x0B	Reset	
0x0C	SetProperty	
0x0D	Reserved	
0x0E	Reserved	
0x0F	Reserved	
0x10	Reserved	
0x11	ConfigureMemory	
0x12	Reserved	
0x13	Reserved	
0x14	Reserved	
0x15	KeyProvision	

Table 211. Response tags

Response tag	Name	Description
0xA0	GenericResponse	The response tag specifies one of the responses the bootloader (target) returns to the host. The valid response tags are listed here.
0xA3	ReadMemoryResponse	
0xA7	GetPropertyResponse (used for sending responses to GetProperty command only)	
0xA3	ReadMemoryResponse (used for sending responses to ReadMemory command only)	
0xAF	FlashReadOnceResponse (used for sending responses to FlashReadOnce command only)	
0xB5	KeyProvisionResponse	

Flags: Each command packet contains a flag byte. Only bit 0 of the flag byte is used. If bit 0 of the flag byte is set to 1, then data packets follow the command sequence. The number of bytes that are transferred in the data phase is determined by a command specific parameter in the parameters array.

ParameterCount: The number of parameters included in the command packet.

Parameters: The parameters are word-length (32 bits). With the default maximum packet size of 32 bytes, a command packet can contain up to seven parameters.

8.5.7 Response packet

The responses are carried using the same command packet format wrapped with framing packet data. Types of responses include:

- GenericResponse.
- GetPropertyResponse.
- ReadMemoryResponse.
- FlashReadOnceResponse.
- KeyProvisionResponse.

GenericResponse: After the bootloader has processed a command, the bootloader sends a generic response with status and command tag information to the host. The generic response is the last packet in the command protocol sequence. The generic response packet contains the framing packet data and the command packet data (with generic response tag = 0xA0) and a list of parameters (defined in the next section). The parameter count field in the header is always set to 2, for status code and command tag parameters.

Table 212. GenericResponse parameters

Byte #	Parameter	Description
0 - 3	Status code	The Status codes are errors encountered during the execution of a command by the target. If a command succeeds, then a kStatus_Success code is returned.
4 - 7	Command tag	The Command tag parameter identifies the response to the command sent by the host.

GetPropertyResponse: The GetPropertyResponse packet is sent by the target in response to the host query that uses the GetProperty command. The GetPropertyResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a GetPropertyResponse tag value (0xA7).

The parameter count field in the header is set to greater than 1, to always include the status code and one or many property values.

Table 213. GetPropertyResponse parameters

Byte #	Value	Parameter
0 - 3		Status code
4 - 7		Property value
. . .		. . .
		Can be up to maximum 6 property values, limited to the size of the 32-bit command packet and property type.

ReadMemoryResponse: The ReadMemoryResponse packet is sent by the target in response to the host sending a ReadMemory command. The ReadMemoryResponse packet contains the framing packet data and the command packet data, with the

command/response tag set to a ReadMemoryResponse tag value (0xA3), the flags field set to kCommandFlag_HasDataPhase (1).

The parameter count set to two for the status code and the data byte count parameters shown below.

Table 214. ReadMemoryResponse parameters

Byte #	Parameter	Description
0 - 3	Status code	The status of the associated Read Memory command.
4 - 7	Data byte count	The number of bytes sent in the data phase.

FlashReadOnceResponse: The FlashReadOnceResponse packet is sent by the target in response to the host sending a FlashReadOnce command. The FlashReadOnceResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a FlashReadOnceResponse tag value (0xAF), and the flags field set to 0. The parameter count is set to 2 plus *the number of words* requested to be read in the FlashReadOnceCommand.

Table 215. FlashReadOnceResponse parameters

Byte #	Value	Parameter
0 - 3		Status code
4 - 7		Byte count to read
. . .		. . .
		Can be up to 20 bytes of requested read data.

The KeyProvisionResponse packet is sent by the target in response to the host sending a KeyProvision command. The KeyProvisionResponse packet contains the framing packet data and command packet data, with the command/response tag set to a KeyProvisionResponse tag value (0xB5), and the flags field set to kCommandFlag_HasDataPhase (1).

Table 216. KeyProvisionResponse parameters

Byte #	Value	Parameter
0 - 3		Status code
4 - 7		Data Byte count

8.6 The bootloader command set

8.6.1 Introduction

All bootloader commands follow the command packet format wrapped by the framing packet as explained in previous sections.

For a list of status codes returned by bootloader see Section 8.6.14 “KeyProvision command (for version 1B only)”.

8.6.2 GetProperty command

The GetProperty command is used to query the bootloader about various properties and settings. Each supported property has a unique 32-bit tag associated with it. The tag

occupies the first parameter of the command packet. The target returns a GetPropertyResponse packet with the property values for the property identified with the tag in the GetProperty command.

Properties are the defined units of data that can be accessed with the GetProperty or SetProperty commands. Properties may be read-only or read-write. All read-write properties are 32-bit integers, so they can easily be carried in a command parameter.

The 32-bit property tag is the only parameter required for GetProperty command.

Table 217. Parameters for GetProperty Command

Byte #	Parameter
0 - 3	Property tag See section 6.6.17 for more details.
4 - 7	External Memory Identifier (only applies to get property for external memory, or status identifier if the property tag equals to 8).

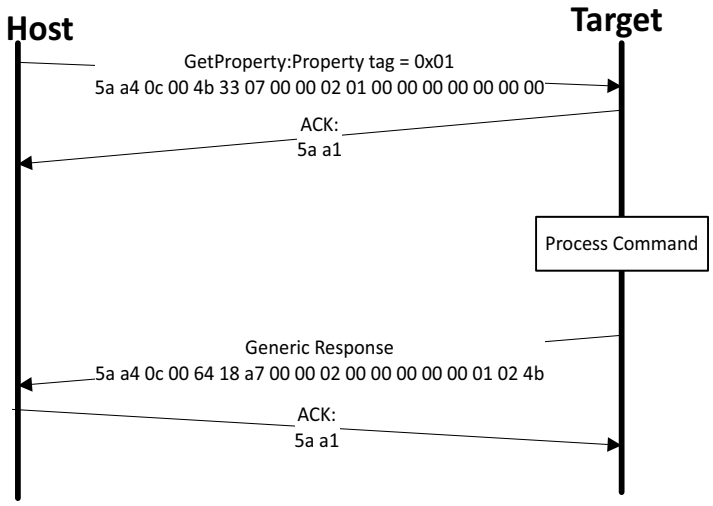


Fig 26. Protocol sequence for GetProperty command

Table 218. GetProperty command packet format (example)

GetProperty	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x0C 0x00
	Crc16	0x4B 0x33
Command packet	CommandTag	0x07 – GetProperty
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x02
	PropertyTag	0x00000001 - CurrentVersion
	Memory ID	0x00000000 - Internal Flash

The GetProperty command has no data phase.

Response: In response to a GetProperty command, the target sends a

GetPropertyResponse packet with the response tag set to 0xA7. The parameter count indicates the number of parameters sent for the property values, with the first parameter showing status code 0, followed by the property value(s). Table 219 shows an example of a GetPropertyResponse packet.

Table 219. GetProperty response packet format (example)

GetPropertyResponse	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x0c 0x00 (12 bytes)
	Crc16	0x07 0x7a
Command packet	ResponseTag	0xA7
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x02
	Status	0x00000000
	PropertyValue	0x0000014b - CurrentVersion

8.6.3 SetProperty command

The SetProperty command is used to change or alter the values of the properties or options of the bootloader. The command accepts the same property tags used with the GetProperty command. However, only some properties are writable. If an attempt to write a read-only property is made, an error is returned indicating the property is read-only and cannot be changed.

The property tag and the new value to set are the two parameters required for the SetProperty command.

Table 220. Parameters for SetProperty command

Byte #	Command
0 - 3	Property tag
4 - 7	Property value

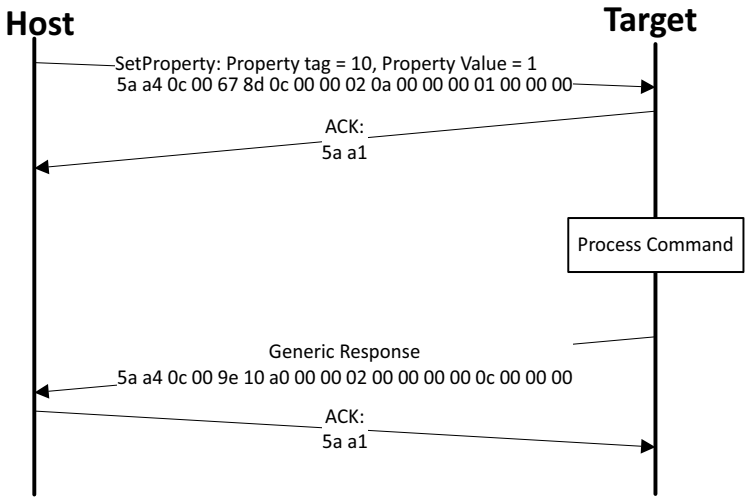


Fig 27. Protocol sequence for SetProperty Command

Table 221. SetProperty command packet format (example)

SetProperty	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x0C 0x00
	Crc16	0x67 0x8D
Command packet	CommandTag	0x0C – SetProperty with property tag 10
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x02
	PropertyTag	0x0000000A - VerifyWrites
	PropertyValue	0x00000001

The SetProperty command has no data phase.

Response: The target returns a GenericResponse packet with one of following status codes:

Table 222. SetProperty response status codes

Status code
kStatus_Success
kStatus_ReadOnly
kStatus_UnknownProperty
kStatus_InvalidArgument

8.6.4 FlashEraseAll command

The FlashEraseAll command performs an erase of the entire flash memory. If any flash regions are protected, then the FlashEraseAll command fails and returns an error status

code. The Command tag for FlashEraseAll command is 0x01 set in the commandTag field of the command packet.

The FlashEraseAll command requires memory ID. If memory ID is not specified, the internal flash (memory ID =0) will be selected as default.

Table 223. Parameter for FlashEraseAll command

Byte #	Parameter	
0-3	Memory ID	
	0x000	Internal Flash
	0x010	Execute-only region in Internal Flash
	0x001	Serial NOR through QuadSPI
	0x008	Parallel NOR through SEMC
	0x009	Serial NOR through FlexSPI
	0x100	SLC Raw NAND through SEMC
	0x101	Serial NAND through FlexSPI
	0x110	Serial NOR/EEPROM through SPI
	0x120	SD through uSDHC
	0x121	eMMC through uSDHC

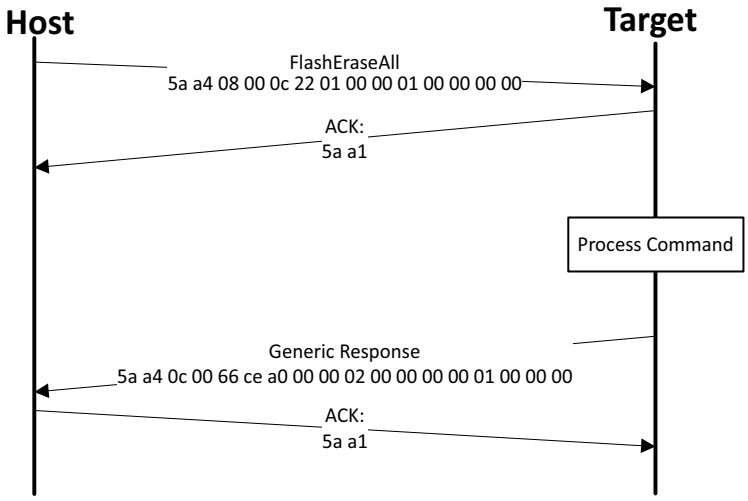


Fig 28. Protocol sequence for FlashEraseAll command

Table 224. FlashEraseAll command packet format (example)

FlashEraseAll	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x08 0x00
	Crc16	0x0C 0x22

Table 224. FlashEraseAll command packet format (example)

FlashEraseAll	Parameter	Value
Command packet	CommandTag	0x01 - FlashEraseAll
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x01
	Memory ID	Refer the above table

The FlashEraseAll command has no data phase.

Response: The target returns a GenericResponse packet with status code either set to kStatus_Success for successful execution of the command or set to an appropriate error status code.

8.6.5 FlashEraseRegion command

The FlashEraseRegion command performs an erase of one or more sectors of the flash memory.

The start address, and number of bytes are the two parameters required for the FlashEraseRegion command. The start and byte count parameters must be 4-byte aligned ([1:0] = 00), or the FlashEraseRegion command fails and returns kStatus_FlashAlignmentError (101). If the region specified does not fit in the flash memory space, the FlashEraseRegion command fails and returns kStatus_FlashAddressError (102). If any part of the region specified is protected, the FlashEraseRegion command fails and returns kStatus_MemoryRangeInvalid (10200).

Table 225. Parameter for FlashEraseRegion command

Byte #	Parameter
0-3	Start address
4 - 7	Byte count
8 - 11	Memory ID

The FlashEraseRegion command has no data phase.

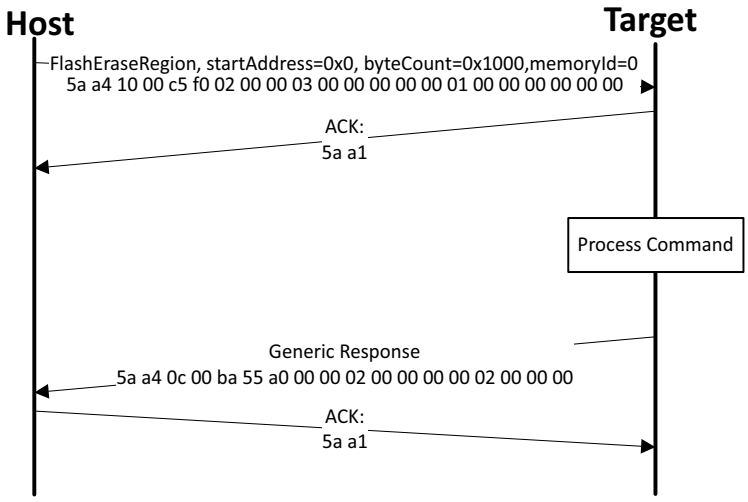


Fig 29. Protocol sequence for FlashEraseRegion command

Response: The target returns a GenericResponse packet with one of the following error status codes.

Table 226. FlashEraseRegion response status codes

Status code
kStatus_Success (0).
kStatus_MemoryRangeInvalid (10200).
kStatus_FlashAlignmentError (101).
kStatus_IFlashAddressError (102).
kStatus_FlashAccessError (103).
kStatus_FlashProtectionViolation (104).
kStatus_FlashCommandFailure (105).

8.6.6 ReadMemory command

The ReadMemory command returns the contents of memory at the given address, for a specified number of bytes. This command can read any region of memory accessible by the CPU and not protected by security.

The start address, and number of bytes are the two parameters required for ReadMemory command. The memory ID is optional. Internal memory will be selected as default if memory ID is not specified.

Table 227. Parameter for read memory command

Byte #	Parameter	Description
0-3	Start address	Start address of memory to read from.
4-7	Byte count	Number of bytes to read and return to caller.
8-11	Memory ID	Internal or external memory Identifier.

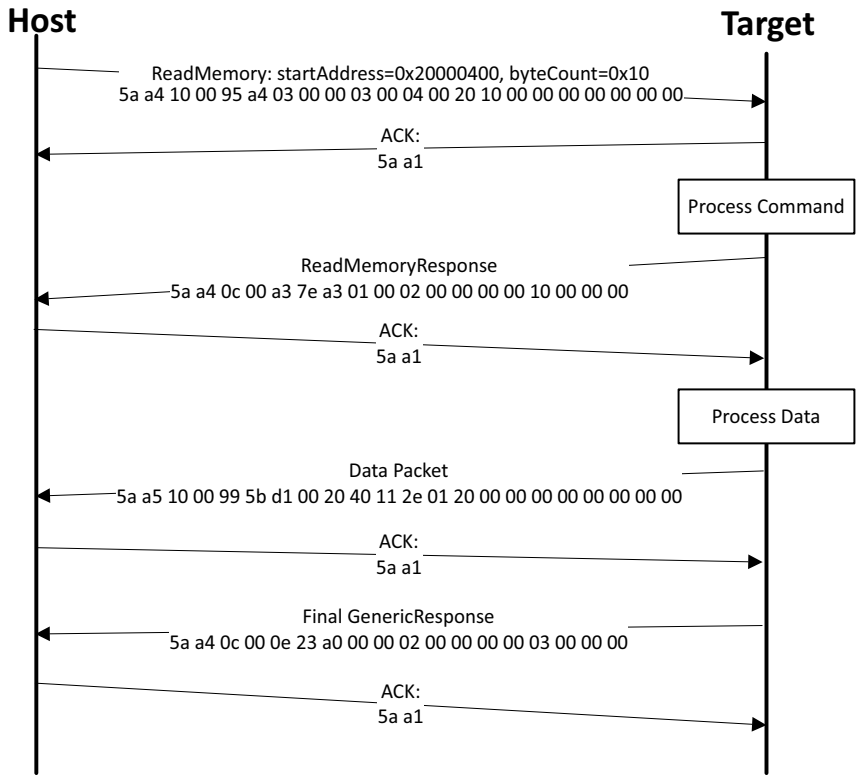


Fig 30. Command sequence for ReadMemory

Table 228. ReadMemory command packet format (example)

ReadMemory	Parameter	Value
Framing packet	Start byte	0x5A
	PpacketType	0xA4, kFramingPacketType_Command
	Length	0x10 0x00
	Crc16	0xF4 0x1B
Command packet	CommandTag	0x03 - ReadMemory
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x03
	StartAddress	0x20000400
	ByteCount	0x00000064
	Memory ID	0x0

Data Phase: The ReadMemory command has a data phase. Because the target works in slave mode, the host needs to pull data packets until the number of bytes of data specified in the byteCount parameter of ReadMemory command are received by host.

Response: The target returns a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command or set to an appropriate error status code

8.6.7 WriteMemory command

The WriteMemory command writes data provided in the data phase to a specified range of bytes in memory (flash or RAM). However, if flash protection is enabled, then writes to protected sectors fail.

Special care must be taken when writing to flash.

- First, any flash sector written to must have been previously erased with a FlashEraseAll or FlashEraseRegion.
- First, any flash sector written to must have been previously erased with a FlashEraseAll or FlashEraseRegion command.
- Writing to flash requires the start address to be page aligned.
- The byte count is rounded up to a page size, and trailing bytes are filled with the flash erase pattern (0xff).
- If the VerifyWrites property is set to true, then writes to flash also performs a flash verify program operation.

When writing to RAM, the start address does not need to be aligned, and the data is not padded.

The start address and number of bytes are the two parameters required for WriteMemory command. The memory ID is optional. Internal memory will be selected as default if memory ID is not specified.

Table 229. Parameters for WriteMemory command

Byte #	Command
0-3	Start address
4-7	Byte count
8-11	Memory ID

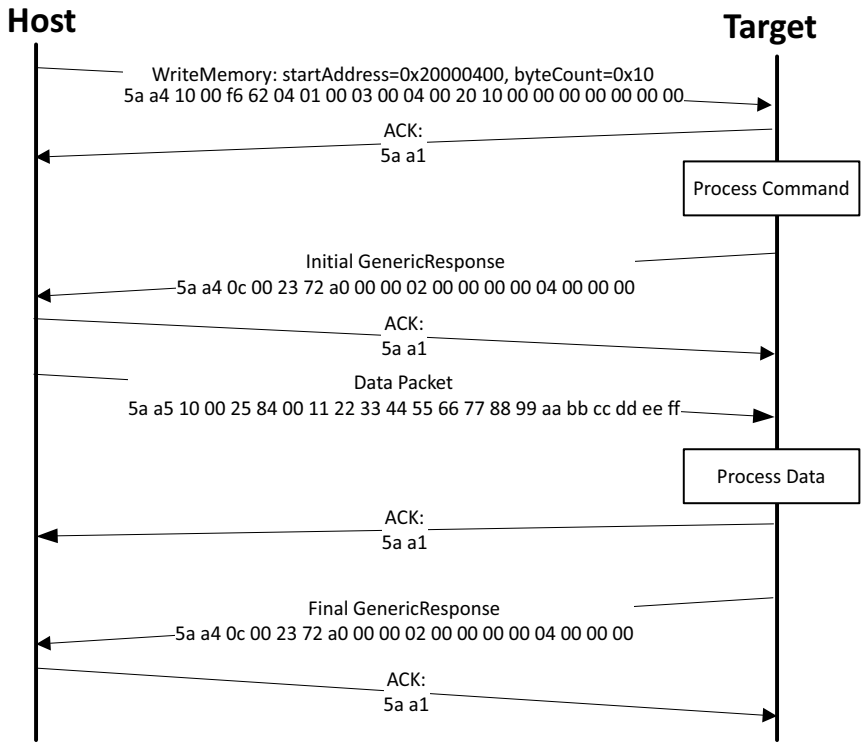


Fig 31. Protocol sequence for WriteMemory command

Table 230. WriteMemory command packet format (example)

WriteMemory	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x10 0x00
	Crc16	0x97 0xDD
Command packet	CommandTag	0x04 - WriteMemory
	Flags	0x01
	Reserved	0x00
	ParameterCount	0x03
	StartAddress	0x20000400
	ByteCount	0x00000064
	Memory ID	0x0

Data Phase: The WriteMemory command has a data phase; the host sends data packets until the number of bytes of data specified in the byteCount parameter of the WriteMemory command are received by the target.

Response: The target returns a GenericResponse packet with a status code set to kStatus_Success upon successful execution of the command, or to an appropriate error status code.

8.6.8 FillMemory command

The FillMemory command fills a range of bytes in memory with a data pattern. It follows the same rules as the WriteMemory command. The difference between FillMemory and WriteMemory is that a data pattern is included in FillMemory command parameter, and there is no data phase for the FillMemory command, while WriteMemory does have a data phase.

Table 231. Parameters for FillMemory command

Byte #	Command
0-3	Start address of memory to fill.
4-7	Number of bytes to write with the pattern • The start address should be 32-bit aligned. • The number of bytes must be evenly divisible by 4. (Note: for a part that uses FTFE flash, the start address should be 64-bit aligned, and the number of bytes must be evenly divisible by 8).
8-11	32-bit pattern.

- To fill with a byte pattern (8-bit), the byte must be replicated four times in the 32-bit pattern.
- To fill with a short pattern (16-bit), the short value must be replicated two times in the32-bit pattern.

For example, to fill a byte value with 0xFE, the word pattern is 0xFEFEFEFE; to fill a short value 0x5AFE, the word pattern is 0x5AFE5AFE.

Special care must be taken while writing to flash.

- First, any flash sector written to must have been previously erased with a FlashEraseAll, or FlashEraseRegion command.
- First, any flash sector written to must have been previously erased with a FlashEraseAll or FlashEraseRegion command.
- Writing to flash requires the start address to be 4-byte aligned ([1:0] = 00).
- If the VerifyWrites property is set to true, then writes to flash also performs a flash verify program operation.

When writing to RAM, the start address does not need to be aligned, and the data is not padded.

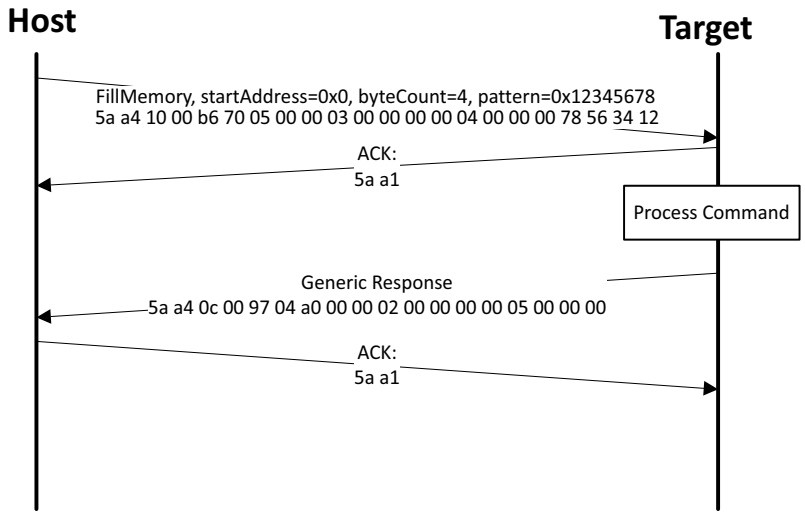


Fig 32. Protocol sequence for FillMemory command

Table 232. FillMemory command packet format (example)

FillMemory	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x10 0x00
	Crc16	0xE4 0x57
Command packet	CommandTag	0x05 – FillMemory
	Flags	0x01
	Reserved	0x00
	ParameterCount	0x03
	StartAddress	0x00007000
	ByteCount	0x00000800
	PatternWord	0x12345678

The FillMemory command has no data phase.

Response: upon successful execution of the command, the target (bootloader) returns a GenericResponse packet with a status code set to kStatus_Success, or to an appropriate error status code.

8.6.9 Execute command

The Execute command results in the bootloader setting the program counter to the code at the provided jump address, R0 to the provided argument, and a Stack pointer to the provided stack pointer address. Prior to the jump, the system is returned to the reset state.

The Jump address, function argument pointer, and stack pointer are the parameters required for the Execute command. If the stack pointer is set to zero, the called code is responsible for setting the processor stack pointer before using the stack.

Table 233. Parameters for Execute command

Byte #	Command
0-3	Jump address.
4-7	Argument word.
8-11	Stack pointer address.

The Execute command has no data phase.

Response: Before executing the Execute command, the target validates the parameters and return a GenericResponse packet with a status code either set to kStatus_Success or an appropriate error status code.

8.6.10 Call command

The Call command executes a function that is written in memory at the address sent in the command. The address needs to be a valid memory location residing in accessible flash (internal or external) or in RAM. The command supports the passing of one 32-bit argument. Although the command supports a stack address, at this time the call still takes place using the current stack pointer. After execution of the function, a 32-bit return value is returned in the generic response message.

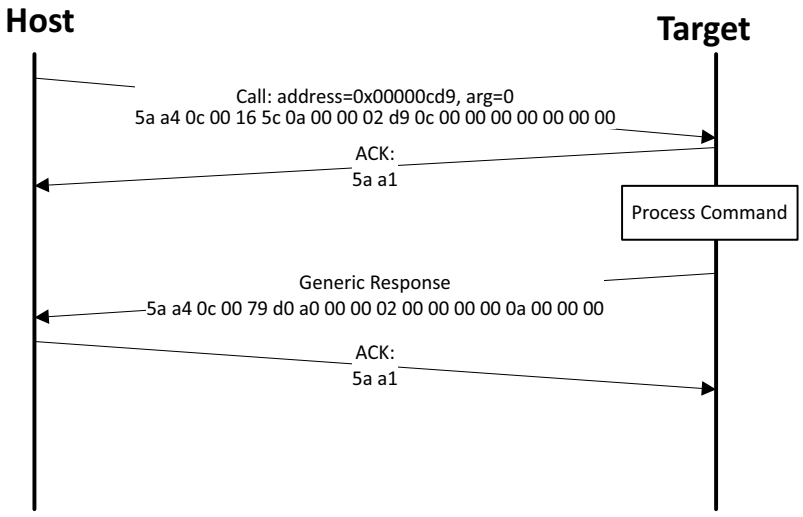


Fig 33. Protocol sequence for call command

Table 234. Parameters for Call command

Byte #	Command
0-3	Jump address
4-7	Argument word
8-11	Stack pointer address

Response: The target returns a GenericResponse packet with a status code either set to the return value of the function called or set to kStatus_InvalidArgument (105).

8.6.11 Reset command

The Reset command results in the bootloader resetting the chip.

The Reset command requires no parameters.

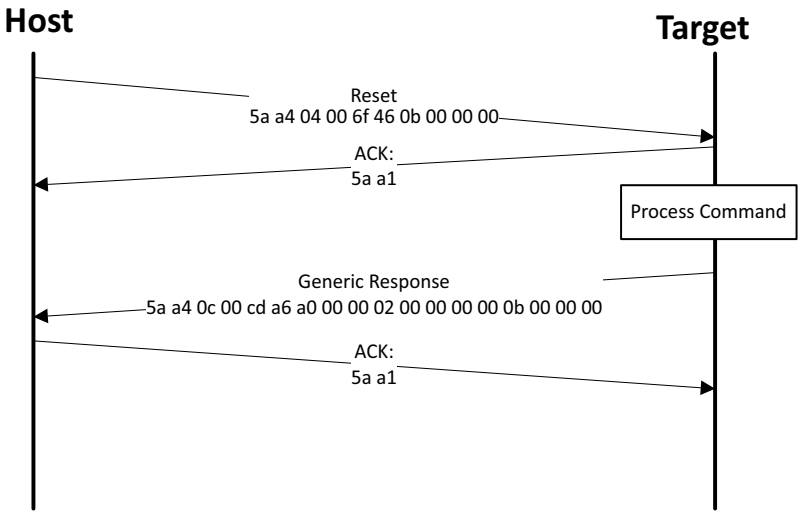


Fig 34. Protocol sequence for Reset command

Table 235. Reset command packet format (example)

Reset	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x04 0x00
	Crc16	0x6F 0x46
Command packet	CommandTag	0x0B - reset
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x03

The Reset command has no data phase.

Response: The target returns a GenericResponse packet with status code set to kStatus_Success, before resetting the chip.

The reset command can also be used to switch boot from flash after successful flash image provisioning via ROM bootloader. After issuing the reset command, allow five seconds for the user application to start running from Flash.

8.6.11.1 Supported Memory IDs (for version 1B only)

The following table shows the supported memory IDs.

Table 236. Supported memory IDs

Memory ID	Description
0	Internal RAM/FLASH (Used for the PRINCE configuration)
0x110	External 1-bit SPI NOR FLASH device

8.6.11.2 1-bit SPI NOR FLASH support (for version 1B only)

The boot ROM support programming 1-bit SPI NOR FLASH devices (which supports 3-byte address read 0x03 or 4-byte address read 0x13) via the following Flash Configuration Option Block.

The boot ROM support either manually configured option or the auto-detected option. When using the first option, users need to specify all the Flash information (Flash size, sector size, page size) in the option block, and when using the 2nd option, the boot ROM is able to detect the Flash information via the read SFDP (0x5A) command, users can set all the Flash information to 0x0s. Users need to be aware that only the device which is JESD216 compliant can support the 2nd option.

Table 237. Serial NOR FLASH configuration option block

Field	Tag	Reserved	Flash info set	Flash size	Sector size	Page size
	[31:28]	[27:16]	[15:12]	[11:8]	[7:4]	[3:0]
option0	0xc		0 - Manual (Select Flash Parameter via Flash Datasheet)	0 - 512KB	0 - 4KB	0 - 256 Bytes
				1 - 1MB	1 - 8KB	1 - 512 Bytes
				2 - 2MB	2 - 32KB	2 - 1KB
			1 - Auto (Detect Flash Parameter via SFDP table)	3 - 4MB	3 - 64KB	3 - 128KB
				4 - 8MB	4 - 128KB	
				5 - 16MB	5 - 256KB	
				6 - 32MB		
				7 - 64MB		
				8 - 64MB		
				9 - 128MB		
				10 - 256MB		

8.6.11.2.1 Example of programming 1-bit SPI NOR FLASH via boot ROM

Use the manually configured parameter.

If the FLASH page size is 256-byte, sector size is 4KB, Flash size is 512KB. The configuration option block is 0xc000_0000 then. Here are the steps to enable the 1-bit SPI NOR FLASH programming using the boot ROM:

```
# Fill the configuration option block to RAM address 0x2000_8000

blhost -u 0x1fc9,0x0021 -- fill-memory 0x20008000 4 0xc0000000

# Enable 1-bit SPI NOR FLASH support using configuration option block stored at
     0x2000_8000

blhost -u 0x1fc9,0x0021 -- configure-memory 0x110 0x20008000

# Erase 64KB starting from address 0
```

```
blhost -u 0x1fc9,0x0021 -- flash-erase-region 0 0x10000 0x110

# Program boot image to the 1-bit SPI NOR FLASH

blhost -u 0x1fc9,0x0021 -- write-memory 0 <boot_image> 0x110
```

8.6.12 ConfigureMemory command

The ConfigureMemory command configures the internal/external memory device using a pre-programmed configuration block. The parameters passed in the command are the memory ID, and then the memory address from which the configuration data can be loaded from. Options for loading the data can be a scenario where the configuration data is written to a RAM or flash location and then this command directs the bootloader to use the data at that location to configure the external memory devices.

Table 238. Parameters for ConfigureMemory command

Byte #	Command
0-3	Memory ID
4-7	Configuration block address

Response: The target (Bootloader) returns a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command or set to an appropriate error code.

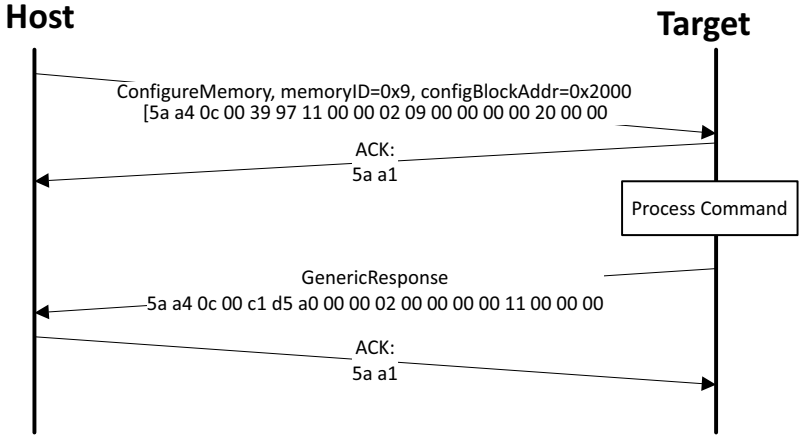


Fig 35. Protocol sequence for ConfigureMemory command

8.6.13 ReceiveSBFile command

The Receive SB File command (ReceiveSbFile) starts the transfer of an SB file to the target. The command only specifies the size in bytes of the SB file that is sent in the data phase. The SB file is processed as it is received by the bootloader. See the Secure boot related sections for more details about the SB file.

Table 239. Parameters for Receive SB File command

Byte #	Command
0-3	Byte count

Data Phase: The Receive SB file command has a data phase; the host sends data packets until the number of bytes of data specified in the byteCount parameter of the Receive SB File command are received by the target.

Response: The target returns a GenericResponse packet with a status code set to the kStatus_Success upon successful execution of the command or set to an appropriate error code.

8.6.14 KeyProvision command (for version 1B only)

The KeyProvision command is a pack of several security related commands, to install pre-shared keys, generate random keys and save them into the Protected Flash Region - Customer Key Store area.

There are three parameters for KeyProvision command, listed in Table 240. The first parameter, <Key Operation> is required to specific the KeyProvision command behavior. The other two parameters, <Key Type> and <Key Size> are required for certain KeyProvision operations.

Table 240. Parameters for KeyProvision command

Byte #	Command
0-3	Key operation
4-7	Key Type / Memory ID (optional for some Key Operations)
8-11	Key Size (optional for some Key Operations)

Table 241 and Table 242 describes the details of each KeyProvision operation and Key Type.

Table 241. KeyProvision operation details

Value	Operation	Details
0	Enroll	Key Provision device enrollment. Generates activation code. For example, PUF key. <Key Type> and <Key Size> are not used for this operation.
1	SetUserKey	Send <Key size> bytes of the <Key Type> key to ROM from host. Incoming data Phase is required to transfer the key bytes.
2	SetIntrinsicKey	Generate <Key Size> bytes of the key specified by <Key Type> in key store
3	WriteNonVolatile	Write the key store in RAM to a nonvolatile memory specified by <Memory ID>. <Key Size> is not used for this operation
4	ReadNonVolatile	Load the key store to RAM from a nonvolatile memory specified by <Memory ID>. <Key Size> is not used.
5	WriteKeyStore	Send the key store to ROM from host. Incoming data Phase is required to transfer the key bytes. Data byte size is fixed as key store size. <Key Type> and <Key Size> are not used for this operation.
6	ReadKeyStore	Read the key store from ROM to host. Outgoing data Phase is required to transfer the key bytes. Data byte size is fixed as key store size. <Key Type> and <Key Size> are not used for this operation.

Table 242. Key Type details

Value	Key Type
0x0	Invalid
0x1	HashCrypt SRK
0x2	OTFAD KEK
0x3	Firmware update key 0
0x4	Firmware update key 1
0x5	Firmware update key 2
0x6	Firmware update key 3
0x7	Firmware update key 0
0x8	Firmware update key 1
0x9	Firmware update key 2
0xA	Firmware update key 3
0xB	User key
0xC	Reserved

Command: KeyProvision command packet format is shown in Table 243.

Table 243. KeyProvision command packet format (example)

KeyProvision	Parameter	Value
Framing packet	start byte	0x5A
	packet type	0xA4, kFramingPacketType_Command
	length	0x10, 0x00
	crc16	0x57, 0x32
Command packet	command tag	0x15
	flags	0x00 (no data phase, 0x01 for has data phase)
	reserved	0x00
	parameter count	0x03
	key operation	0x00000002 (see Table 241)
	key type / memory ID	0x00000000 (see Table 242)
	key size	0x00000100

Data Phase: It is determined by <Key Operation> based on the Incoming or outgoing data phase of the KeyProvision command.

For an incoming packet, the host sends data packets until the number of data bytes is specified by <Key Size> or the key store size are received by the target.

For outgoing data phase, the host needs to pull data packets until it receives the entire key store data bytes. The key store size is sent to the host by KeyProvision response.

Response: The target returns a GenericResponse packet for the key operations without data phase, such as Enroll. It returns a KeyProvisionResponse packet for the other key operations, such as WriteKeyStore.

For the GenericResponse, see Section 8.5.7 “Response packet”.

Table 244 describes the KeyProvisionResponse packet.

Table 244. KeyProvision response packet format (Example)

KeyProvision	Parameter	Value
Framing packet	start byte	0x5A
	packet type	0xA4, kFramingPacketType_Command
	length	0x10, 0x00
	crc16	0xXX, 0xXX
Command packet	command tag	0x15
	flags	0x01 (has data phase)
	reserved	0x00
	parameter count	0x02
	status	0x00000000
	key size	0x00000100

8.6.14.1 Get/SetProperty command properties

This section lists the properties of the GetProperty and SetProperty commands.

Table 245. Properties used by Get/SetProperty commands, sorted by values

Property	Writable	Tag Value	Size	Description
CurrentVersion	No	01h	4	Current bootloader version.
AvailablePeripherals	No	02h	4	The set of peripherals supported on this chip.
FlashStartAddress	No	03h	4	Start address of program flash.
FlashSizeInBytes	No	04h	4	Size in bytes of program flash.
AvailableCommands	No	07h	4	The set of commands supported by the bootloader
Check Status	No	08h	4	Return the status based on specified status identifier 0 - CRC status 32-bit return value for CRC Check 10401 - Application CRC check failed 10402 - Application CRC check is inactive 10403 - Application CRC check is invalid 1 - Last Error See the details of last error in later section

Table 245. Properties used by Get/SetProperty commands, sorted by values *…continued*

Property	Writable	Tag Value	Size	Description
MaxPacketSize	No	0Bh	4	Maximum supported packet size for the currently active peripheral interface.
ReservedRegions	No	0Ch	8*n	List of memory regions reserved by the bootloader. Returned as value pairs (<start-address-of-region>, <end-address-of-region>). If HasDataPhase flag is not set, then the Response packet parameter count indicates the number of pairs. If HasDataPhase flag is set, then the second parameter is the number of bytes in the data phase. "n" indicates number of memory region pairs
LifeCycleState	No	11h	4	Indicates whether Flash security is enabled 0x5aa55aa5 - Device is in development lifecyle 0xc33cc33c - Device is in deployment lifecycle

Table 245. Properties used by Get/SetProperty commands, sorted by values *…continued*

Property	Writable	Tag Value	Size	Description
UniqueDevice/UUID	No	12h	16	Unique device identification
ExternalMemoryAttributes	No	19h	24	List of attributes supported by the specified memory Id (0x110=SPI NOR FLASH). See description for the return value in the section ExternalMemoryAttributes Property
IrqNotifierPin	Yes	1ch	4	IRQ Notifier Pin setting bit[7:0] - gpio pin bit[15:8] - gpio port bit [30:16] - reserved bit [31] - enable flag, 0 - disable, 1 - enable

8.6.14.1.1 Property definitions

Get/Set property definitions are provided in this section.

CurrentVersion property: The value of this property is a 4-byte structure containing the current version of the bootloader.

Table 246. CurrentVersion property fields

Bit	[31:24]	[23:16]	[15:8]	[7:0]
Field	Name = 'K' (0x4B)	Major version	Major version	Bugfix version

AvailablePeripherals property: The value of this property is a bitfield that lists the peripherals supported by the bootloader and the hardware on which it is running.

Table 247. Peripheral bits

Bit	[31:7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Field	Reserved	Reserved	Reserved	USB HID	Reserved	SPI Slave	I2C Slave	LPUART

If the peripheral is available, then the corresponding bit will be set in the property value. All reserved bits must be set to 0.

AvailableCommands property: This property value is a bitfield with set bits indicating the commands enabled in the bootloader. Only commands that can be sent from the host to the target are listed in the bitfield. Response commands such as GenericResponse are excluded.

The bit number that identifies whether a command is present is the command's tag value minus 1. 1 is subtracted from the command tag because the lowest command tag value is 0x01. To get the bit mask for a given command, use this expression: mask = 1 << (tag - 1).

Table 248. Command bits

Bit	[Others]	[20]	[16]	[15]	[14]	[13]	[12]	[11]	[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Command	Reserved	KeyProvisioning	ConfigureMemory	FlashReadResource	FlashReadOne	FlashProgramOne	FlashEraseAllUnsecure	SetProperty	Reset	Call	Execute	ReceiveSBFile	GetProperty	FlashSecurityDisable	FillMemory	WriteMemory	ReadMemory	FlashEraseRegion	FlashEraseAll

ExternalMemoryAttributes property: The value returned by this property is a 24-byte data structure containing available external memory attributes: start address, total size in KB, page size, sector size, and block size. Below is the breakdown of 24-byte structure.

Table 249. Fields of ExternalMemoryAttributes property

Field offset	Field Description
0 - 3	The value returned is a bitmap showing the supported attributes for the external memory, with the corresponding bitfield set. 0x00000001 - start address 0x00000002 - total size 0x00000004 - page size 0x00000008 - sector size 0x00000010 - block size
4 - 7	Start address of external memory.
8 - 11	Total size of external memory in kilobytes.
12 - 15	Page size of external memory in bytes.
16 - 19	Sector size of external memory in bytes.
20 - 23	Block size of external memory in bytes.

GetLastError property: The following table lists the response words and corresponding error conditions.

Table 250. Response word and error description

Reponse word	Error	Description
0x0b37f300	kLog_Auth_CrcCheck_Fail	CRC checksum is wrong.
0x0b35f300	kLog_Auth_ImageEntryCheck_Fail	Application entry point is invalid or stack address is invalid.
0x0b38f300	-	Reserved.
0x0d70f300	kLog_Tzm_DeviceMode_Fail	
0x0d71f300	kLog_Tzm_FusesMode_Fail	
0x0d72f300	kLog_Tzm_ImageMode_Fail	
0x0c00f500	kLog_Jump_Fail_Fatal	Failed to jump to application.
0x0602f30<n>		Passive boot failed.
0x0702f30<n>	kLog_Recoveryboot_Fail_Reason	Recovery boot failed.

8.7 Bootloader Status Error Codes

This section describes the status error codes that the Bootloader returns to the host.

Table 251. Bootloader status error codes, sorted by value

Error Code	Value	Description
kStatus_Success	0	Operation succeeded without error.
kStatus_Fail	1	Operation failed with a generic error.
kStatus_ReadOnly	2	Request value cannot be changed because it is read-only.
kStatus_OutOfRange	3	Requested value is out of range.
kStatus_InvalidArgument	4	The requested command's argument is undefined.
kStatus_Timeout	5	A timeout occurred.
kStatus_NoTransferInProgress	6	No send in progress.
kStatus_FLASH_Success	0	API is executed successfully.
kStatus_FLASH_InvalidArgument	4	An invalid argument is provided.
kStatus_FlashSizeError	100	Not used.
kStatus_FlashAlignmentError	101	Address or length does not meet the required alignment.
kStatus_FlashAddressError	102	Address or length is outside of addressable memory.
kStatus_FlashAccessError	103	The FTFA_FSTAT[ACCERR] bit is set.
kStatus_FlashProtectionViolation	104	The FTFA_FSTAT[FPVIOL] bit is set.
kStatus_FlashCommandFailure	105	The FTFA_FSTAT[MGSTAT0] bit is set.
kStatus_FlashUnknownProperty	106	Unknown Flash property.
kStatus_FlashEraseKeyError	107	The provided key does not match the programmed Flash key.
kStatus_FlashRegionExecuteOnly	108	The area of Flash is protected as execute-only.
kStatus_FLASH_ExecuteInRamFunctionNotReady	109	Execute-in-RAM function is not available.
kStatus_FLASH_CommandNotSupported	111	Flash API is not supported.
kStatus_FLASH_ReadOnlyProperty	112	The Flash property is read-only.
kStatus_FLASH_InvalidPropertyValue	113	The Flash property value is out of range.
kStatus_FLASH_InvalidSpeculationOption	114	The Flash prefetch speculation option is invalid.
kStatus_FLASH_EccError	116	An error was generated during command execution that may, or may not be correctable.
kStatus_FLASH_CompareError	117	Destination and source memory contents do not match.
kStatus_FLASH_RegulationLoss	118	A loss of regulation occurred during read operation.
kStatus_FLASH_InvalidWaitStateCycles	119	The wait state cycle set to R/W mode is invalid.
kStatus_FLASH_OutOfDateCfpaPage	132	CFPA page version is out of date.
kStatus_FLASH_BlankIfrPageData	133	Blank page cannot be read.
kStatus_FLASH_EncryptedRegionsEraseNotDoneAtOnce	134	Encrypted Flash subregions are not erased at once.
kStatus_FLASH_ProgramVerificationNotAllowed	135	Program verification is not allowed when the encryption is enabled.
kStatus_FLASH_HashCheckError	136	Hash check of page data has failed.
kStatus_FLASH_SealedFfrRegion	137	The FFR region is sealed.
kStatus_FLASH_FfrRegionWriteBroken	138	The FFR spec region is not allowed to be written discontinuously.

Table 251. Bootloader status error codes, sorted by value *…continued*

Error Code	Value	Description
kStatus_FLASH_NmpaAccessNotAllowed	139	The NMPA region is not allowed to be read/written/erased.
kStatus_FLASH_CmpaCfgDirectEraseNotAllowed	140	The CMPA Cfg region is not allowed to be erased directly.
kStatus_FLASH_FfrBankIsLocked	141	The FFR bank region is locked.
kStatus_FLASH_CfpaScratchPageInvalid	148	CFPA Scratch Page is invalid.
kStatus_FLASH_CfpaVersionRollbackDisallowed	149	CFPA version rollback is not allowed.
kStatus_UnknownCommand	10000	Command is not recognized.
kStatus_SecurityViolation	10001	Security violation happened when receiving disallowed commands.
kStatus_AbortDataPhase	10002	Sender requested data phase abort.
kStatus_Ping	10003	Ping command received from the host.
kStatus_CommandUnsupported	10006	Unsupported command received.
kStatusRomLdrSectionOverrun	10100	Reached end of the SSB file processing.
kStatusRomLdrSignature	10101	Incorrect signature or version.
kStatusRomLdrSectionLength	10102	The bootOffset/ new section count is out of range.
kStatusRomLdrUnencryptedOnly	10103	The non-encrypted image is disabled.
kStatusRomLdrEOFReached	10104	The end of the image file has been reached.
kStatusRomLdrChecksum	10105	Checksum for command tag block is invalid.
kStatusRomLdrCrc32Error	10106	The CRC-32 of the data for a load command is incorrect.
kStatusRomLdrUnknownCommand	10107	An unknown command was detected in the SB file.
kStatusRomLdrIdNotFound	10108	No bootable section found in SB file.
kStatusRomLdrDataUnderrun	10109	The SB state machine is waiting for more data.
kStatusRomLdrJumpReturned	10110	The function that was jumped to by the SB file has returned.
kStatusRomLdrCallFailed	10111	The call command in the SB file has failed.
kStatusRomLdrKeyNotFound	10112	A matching key was not found in the SB file's key dictionary to unencrypt the section.
kStatusRomLdrSecureOnly	10113	The SB file is unencrypted and security on the target is disabled.
kStatusRomLdrResetReturned	10114	The SB file reset operation has unexpectedly returned.
kStatusRomLdrRollbackBlocked	10115	An image version rollback event has been detected.
kStatusRomLdrInvalidSectionMacCount	10116	Invalid Section MAC count detected in the SB file.
kStatusRomLdrUnexpectedCommand	10117	The command tag in the SB file is unexpected.
kStatusRomLdrBadSBKEK	10118	Bad SBKEK detected.
kStatusMemoryRangeInvalid	10200	The requested address range does not match an entry, or the length extends past the matching entry's end address.
kStatusMemoryReadFailed	10201	Memory read failed.
kStatusMemoryWriteFailed	10202	Memory write failed.
kStatusMemoryCumulativeWrite	10203	Cumulative write occurred due to write to an unerased Flash region.
kStatusMemoryNotConfigured	10205	Memory not configured prior to access.
kStatusMemoryAlignmentError	10206	Alignment error occurred during memory access.
kStatusMemoryVerifyFailed	10207	Verification operation failed after programming/erasing Flash.

Table 251. Bootloader status error codes, sorted by value *…continued*

Error Code	Value	Description
kStatusMemoryWriteProtected	10208	The memory being written to is write protected.
kStatusMemoryAddressError	10209	Invalid or wrong memory address has been specified.
kStatusMemoryBlankCheckFailed	10210	Check of blank memory status has failed.
kStatusMemoryBlankPageReadDisallowed	10211	Memory is blank and read command is not allowed.
kStatusMemoryProtectedPageReadDisallowed	10212	Memory is protected and read command is not allowed.
kStatusMemoryFfrSpecRegionWriteBroken	10213	The write operation to the FFR region was broken.
kStatusMemoryUnsupportedCommand	10214	The memory command is not supported.
kStatus_UnknownProperty	10300	The requested property value is undefined.
kStatus_ReadOnlyProperty	10301	The requested property value cannot be written.
kStatus_InvalidPropertyValue	10302	The specified property value is invalid.
kStatus_AppCrcCheckPassed	10400	CRC check is valid and has passed successfully.
kStatus_AppCrcCheckFailed	10401	CRC check is valid but has failed.
kStatus_AppCrcCheckInactive	10402	CRC check is inactive.
kStatus_AppCrcCheckInvalid	10403	CRC check is invalid because the BCA is invalid, or the CRC parameters are not set (all 0xFF bytes).
kStatus_AppCrcCheckOutOfRange	10404	CRC check is valid, but addresses are out of range.
kStatus_RomApiExecuteCompleted	0	ROM successfully completed processing of SB file/boot image.
kStatus_RomApiNeedMoreData	10801	ROM requires more data to continue processing the boot image.
kStatus_RomApiBufferSizeNotEnough	10802	User buffer is not large enough for Kboot during execution of the specified operation.
kStatus_RomApiInvalidBuffer	10803	User buffer is not appropriately prepared for the sbloader or authentication.
kStatus_SerialNorEepromAddressInvalid	20700	The serial nor eeprom address is invalid.
kStatus_SerialNorEepromTransferError	20701	An Error occurred during the serial nor eeprom transfer.
kStatus_SerialNorEepromTypeInvalid	20702	The serial nor eeprom type is invalid.
kStatus_SerialNorEepromSizeInvalid	20703	The serial nor eeprom size is invalid.
kStatus_SerialNorEepromCommandInvalid	20704	The serial nor eeprom command is invalid.

8.8 UART ISP

8.8.1 Introduction

The bootloader integrates an autobaud detection algorithm for the UART peripheral, thereby providing flexible baud rate choices.

Autobaud feature: If UART*n* is used to connect to the bootloader, then the UART*n*_RX pin must be kept high and not left floating during the detection phase in order to comply with the autobaud detection algorithm. After the bootloader detects the Ping packet (0x5A 0xA6) on UART*n*_RX, the bootloader firmware executes the autobaud sequence.

If the baudrate is successfully detected, then the bootloader sends a Ping packet response [(0x5A 0xA7), protocol version (4 bytes), protocol version options (2 bytes) and crc16 (2 bytes)] at the detected baudrate. The bootloader then enters a loop, waiting for bootloader commands via the UART peripheral.

NOTE: The data bytes of the ping packet must be sent continuously (with no more than 80 ms between bytes) in a fixed UART transmission mode (8-bit data, no parity bit and 1 stop bit). If the bytes of the ping packet are sent one-by-one with more than 80 ms delay between them, then the autobaud detection algorithm may calculate an incorrect baud rate. In this instance, the autobaud detection state machine should be reset.

Supported baud rates: The baud rate is closely related to the MCU core and system clock frequencies. Typical baud rates supported are 9600, 19200, 38400, 57600, 115200 ,230400, 460800 and 1000000.

Packet transfer: After autobaud detection succeeds, bootloader communications can take place over the UART peripheral. The following flow charts show:

- How the host detects an ACK from the target.
- How the host detects a ping response from the target.
- How the host detects a command response from the target.

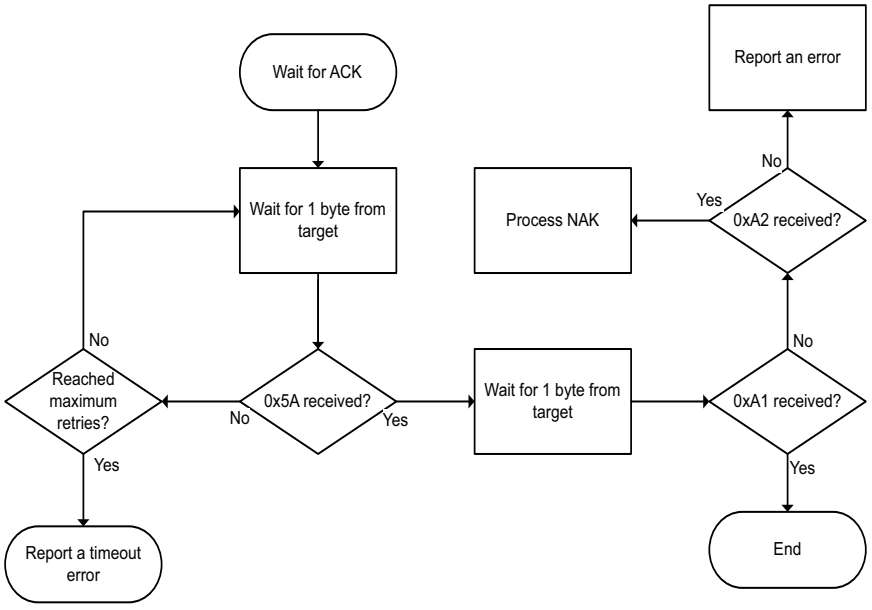


Fig 36. Host reads an ACK from target via UART

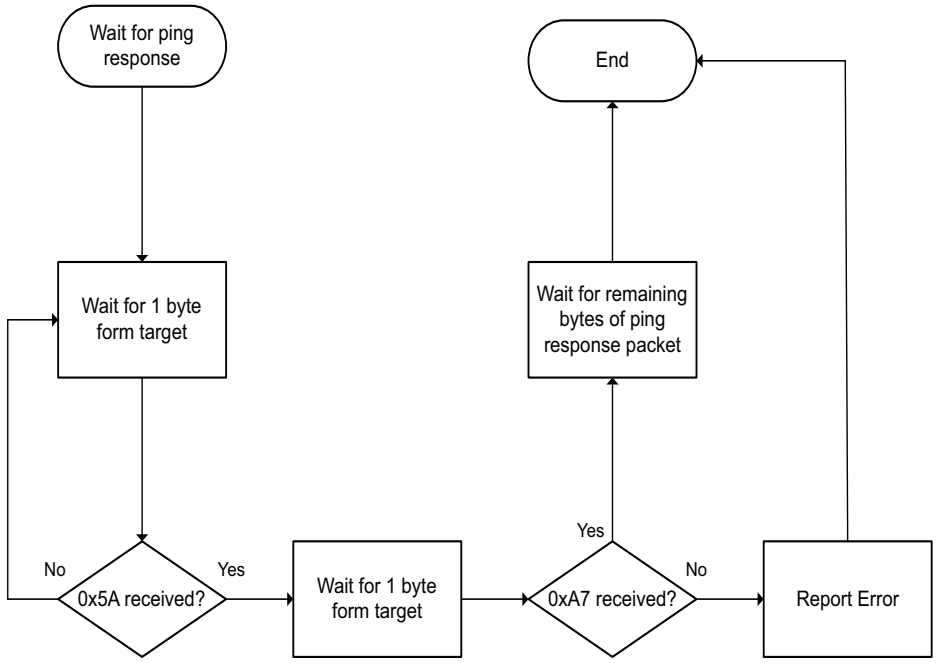


Fig 37. Host reads a ping response from target via UART

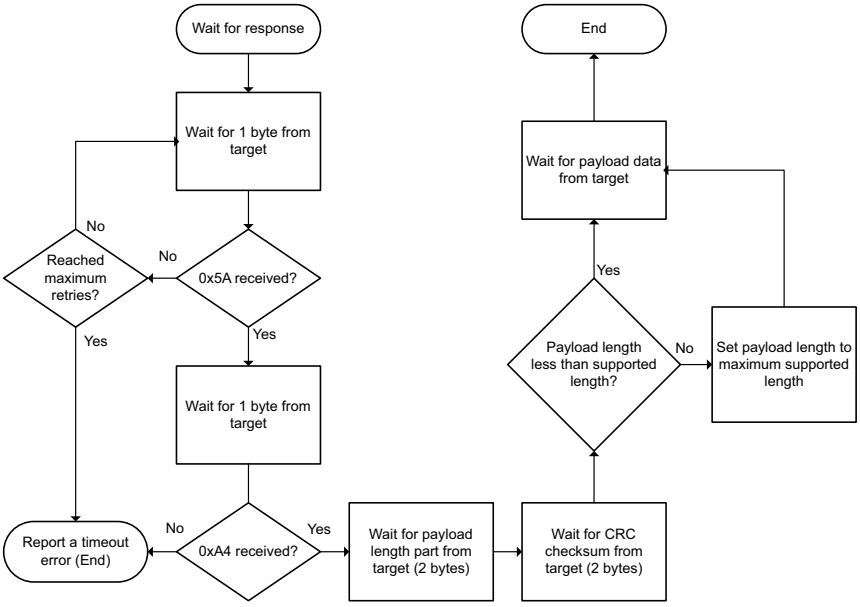


Fig 38. Host reads a command response from target via UART

8.8.2 UART ISP command format

See Section 8.5 “Bootloader packet types” for more details.

8.8.3 UART ISP response format

See Section 8.5 “Bootloader packet types” for more details.

8.8.4 UART ISP data format

See Section 8.5 “Bootloader packet types” for more details.

8.8.5 UART ISP commands

See Section 8.5 “Bootloader packet types” for more details.

8.9 I²C In-System Programming

8.9.1 Introduction

The bootloader supports loading data into flash via the I²C peripheral, where the I²C peripheral serves as the I²C slave. A 7-bit slave address is used during the transfer. The bootloader uses 0x10 as the I²C slave address and supports up to 400 kbit/s as the I²C baud rate.

The maximum supported I²C baud rate depends on the core clock frequency when the bootloader is running. The typical baud rate is 400 kbit/s with factory settings.

Because the I²C peripheral serves as an I²C slave device, each transfer should be started by the host, and each outgoing packet should be fetched by the host.

- An incoming packet is sent by the host with a selected I²C slave address and the direction bit is set to write.
- An outgoing packet is read by the host with a selected I²C slave address and the direction bit is set as read.
- 0x00 is sent as the response to host if the target is busy with processing or preparing data.

The following charts show the communication flow of the host reading the ping and ACK packets, and the corresponding responses from the target.

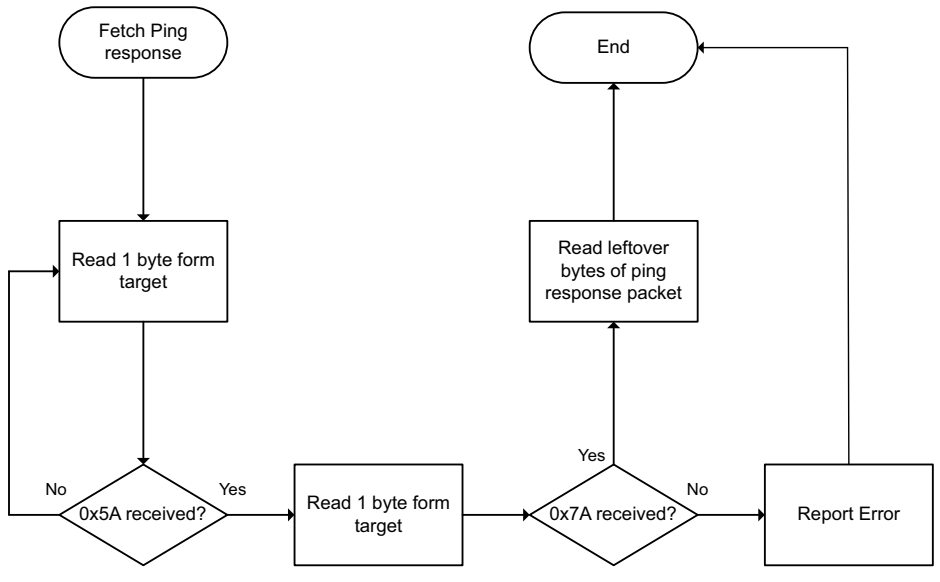


Fig 39. Host reads ping response from target via I²C

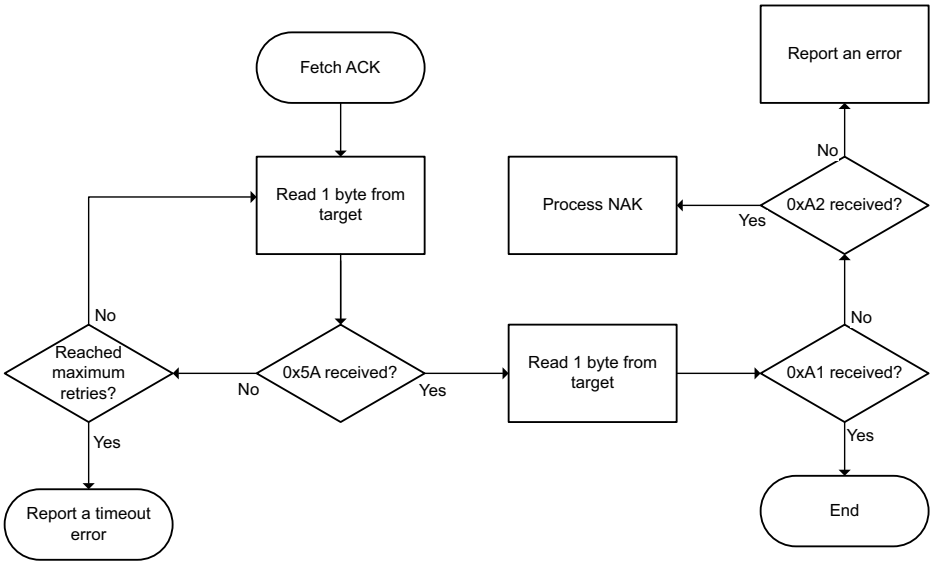


Fig 40. Host reads ACK packet from target via I^2C

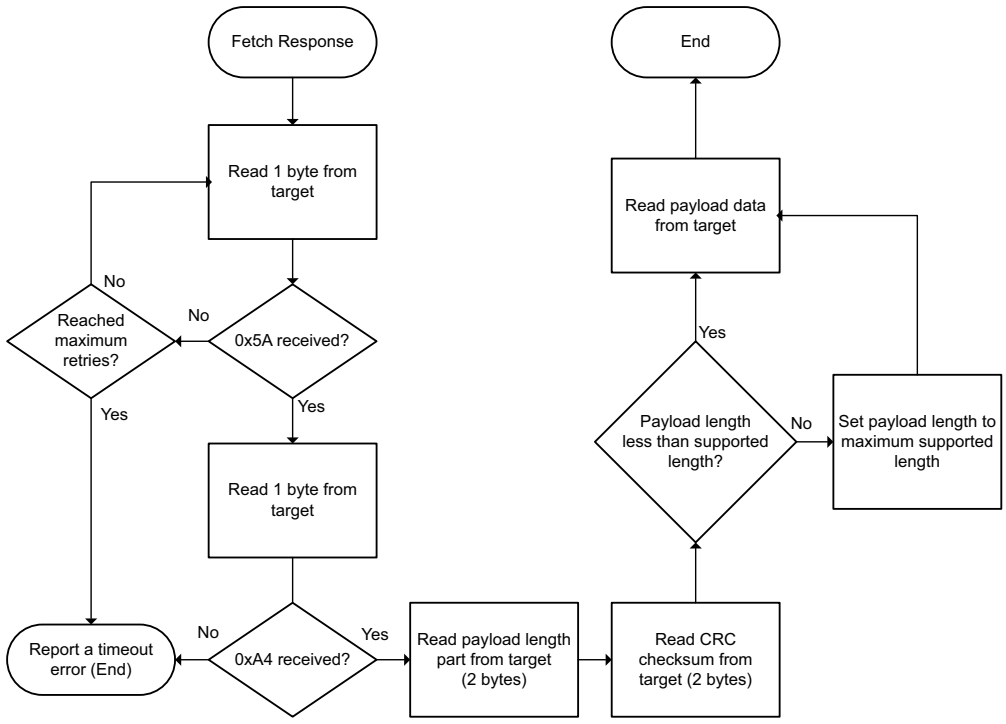


Fig 41. Host reads response from target via I²C

Note: The I2C master needs to read back each byte from the slave in 20ms, otherwise the ROM will abort the transfer (applies to device revision 1B only).

8.9.2 I²C ISP command format

See Section 8.5 “Bootloader packet types” for more details.

8.9.3 I²C ISP response format

See Section 8.5 “Bootloader packet types” for more details.

8.9.4 I²C ISP data format

See Section 8.5 “Bootloader packet types” for more details.

8.9.5 I²C ISP commands

See Section 8.6 “The bootloader command set” for more details.

8.10 SPI In-System programming

8.10.1 Introduction

The bootloader supports loading data into flash via the SPI peripheral, where the SPI peripheral serves as an SPI slave. The SPI transfer should be SPI Mode 3 with 8 data bits.

The maximum supported baud rate of the SPI depends on the core clock frequency when the bootloader is running. The typical baud rate is 2000 kbit/s with the factory settings. The actual baud rate is lower or higher than 2000 kbit/s, depending on the actual value of the core clock.

Because the SPI peripheral serves as an SPI slave device, each transfer should be started by the host, and each outgoing packet should be fetched by the host.

- The transfer on SPI is slightly different from I^2C:
- The host receives 1 byte after it sends out any byte.
- Received bytes should be ignored when the host is sending out bytes to the target
- The host starts reading bytes by sending 0x00s to target

  The byte 0x00 is sent as a response to host if the target is under the following conditions:
  - Processing incoming packet.
  - Preparing outgoing data.
  - Received invalid data.

The bootloader also supports the active notification pin (nIRQ pin) to notify the host processor it is busy or ready for new commands/data. See below figure for the typical physical connection between the host and the bootloader device.

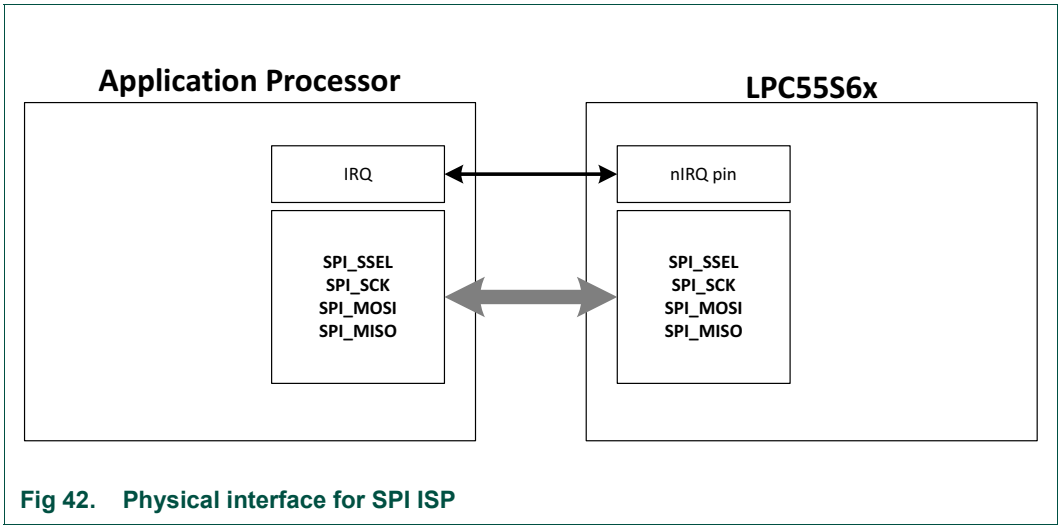


Fig 42. Physical interface for SPI ISP

The following flowcharts show how the host reads a ping response, an ACK and a command response from target via SPI without the nIRQ pin enabled.

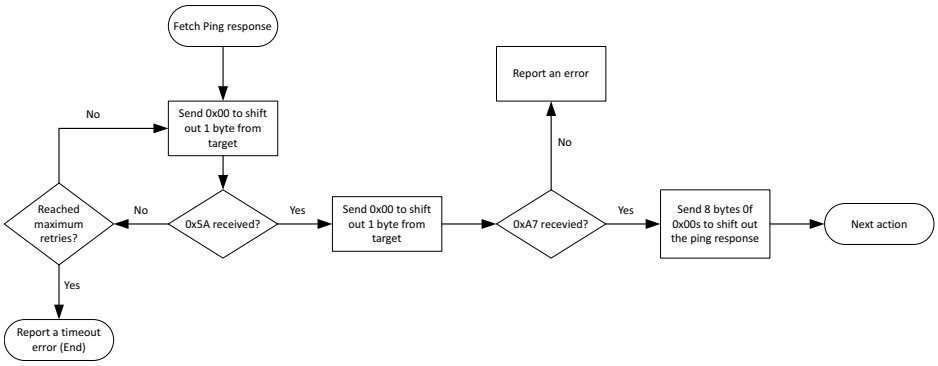


Fig 43. Host reads ping packet from target via SPI

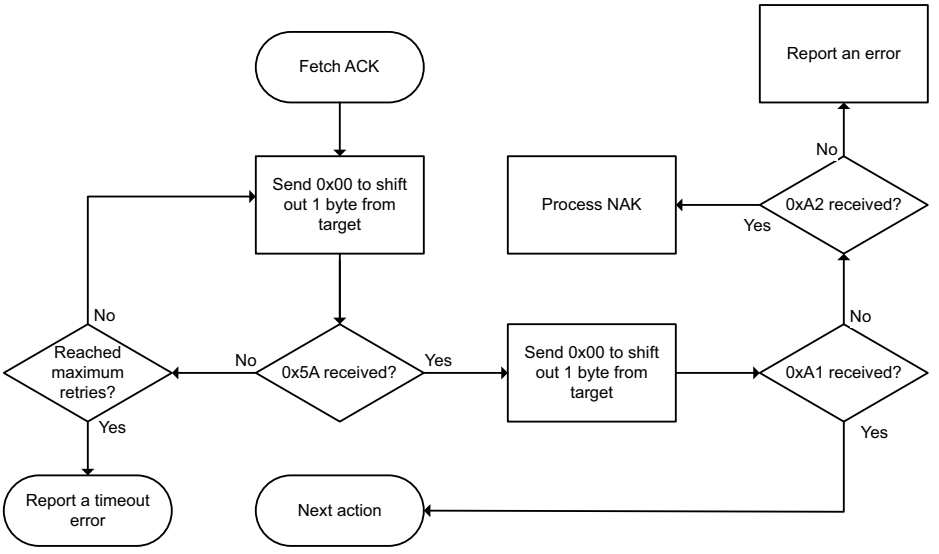


Fig 44. Host reads ACK from target via SPI

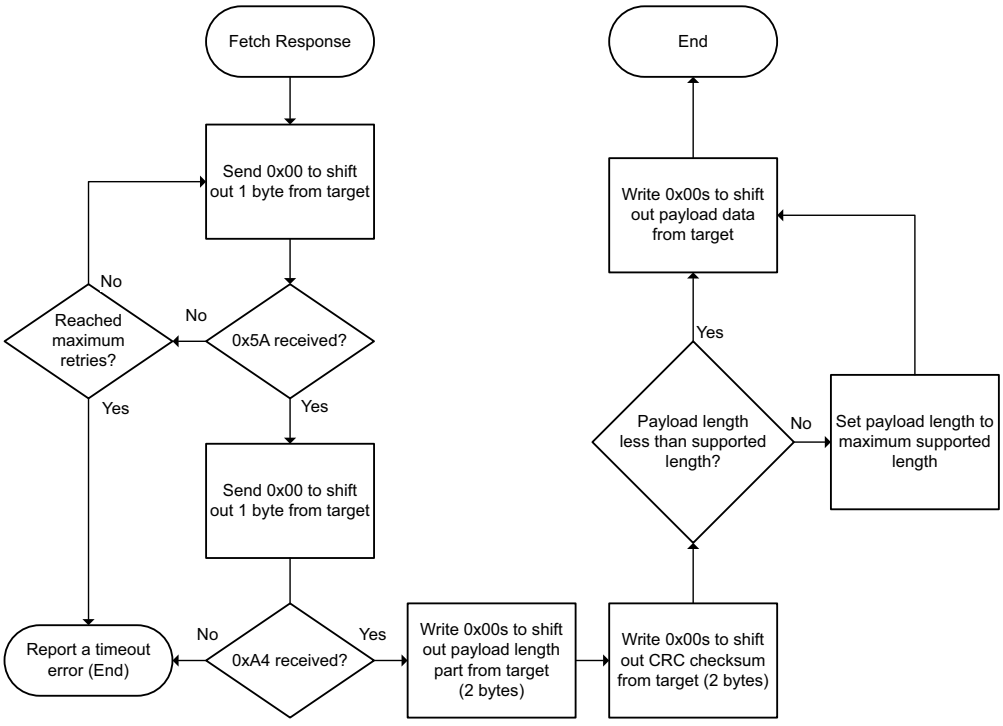


Fig 45. Host reads response from target via SPI

To accelerate the SPI transfer between the host and the bootloader, the bootloader provides an active notification pin known as the nIRQ pin, it can be enabled by the SetProperty command. Once being enabled, the host needs to wait until it sees a negative edge on the nIRQ pin before reading any data from the bootloader, and it needs to wait until the nIRQ pin is high before sending any data to the bootloader.

Note: The SPI master needs to read back each byte in ACK/Response from the slave in 20ms, otherwise the ROM will abort the transfer.

8.10.2 SPI ISP command format

See Section 8.5 “Bootloader packet types” for more details

8.10.3 SPI ISP response format

See Section 8.5 “Bootloader packet types” for more details

8.10.4 SPI ISP data format

See Section 8.5 “Bootloader packet types” for more details

8.10.5 SPI ISP commands

See Section 8.6 “The bootloader command set” for more details

8.11 USB In-System Programming

8.11.1 Introduction

The bootloader supports In-System Programming using the USB peripheral. The target is implemented as USB-HID device classes.

When transfer data through USB-HID device class, USB-HID does not use framing packets. Instead, the packetization, inherent in the USB protocol itself is used. The ability for the device to NAK Out transfers (until they can be received) provides the required flow control. The built-in CRC of each USB packet provides the required error detection

8.11.1.1 Device descriptor

The bootloader configures the default USB VID/PID/Strings as below:

Default VID/PID:

- VID = 0x1FC9.
- PID = 0x0021.

Default Strings:

- Manufacturer [1] = "NXP SEMICONDUCTOR INC".
- Product [2] = "USB COMPOSITE DEVICE".

The USB VID, PID can be customized using the CMPA of the flash. The USB VID and PID can be customized by writing the new VID to the usbVid field and the new PID to the usbPid field of the CMPA in flash.

8.11.1.2 Endpoints

The HID peripheral uses three endpoints:

- Control (0).
- Interrupt IN (1).
- Interrupt OUT (2).

The Interrupt OUT endpoint is optional for HID class devices, but the MCU bootloader uses it as a pipe, where the firmware can NAK send requests from the USB host.

8.11.1.3 HID Reports

There are four HID reports defined and used by the bootloader USB HID peripheral. The report ID determines the direction and type of packet sent in the report; otherwise, the contents of all reports are the same.

Table 252. HID reports assigned for the bootloader

Report ID	Packet Type	Direction
1	Command	OUT
2	Data	OUT
3	Command	IN
4	Data	IN

Each report has a maximum size of 60 bytes. The maximum payload size is 56 bytes. In addition, there is a 4-byte report header that indicates the length (in bytes) of the payload and report id sent the packet.

Note: In the future, the maximum report size may be increased, to support transfers of larger packets. Alternatively, additional reports may be added with larger maximum sizes.

The actual data sent in all of the reports looks like:

Table 253. Data format sent in USB HID packet

0	Report ID
1	Padding
2	Packet Length LSB
3	Packet Length MSB
4	Packet[0]
5	Packet[1]
6	Packet[2]
	...
N+4-1	Packet[N-1]

This data includes the Report ID, which is required if more than one report is defined in the HID report descriptor. The actual data sent and received has a maximum length of 35 bytes. The Packet Length header is written in little-endian format, and it is set to the size (in bytes) of the packet sent in the report. This size does not include the Report ID or the Packet Length header itself. During a data phase, a packet size of 0 indicates a data phase abort request from the receiver.

8.11.2 USB ISP command format

See Section 8.5.6 “Command packet” for more details

8.11.3 USB ISP response format

See Section 8.5.6 “Command packet” for more details

8.11.4 USB ISP data format

See Section 8.5.7 “Response packet” for more details

8.11.5 USB ISP commands

See Section 8.6 “The bootloader command set” for more details

8.12 In-Application-Programming

See Chapter 9 “LPC55S6x/LPC55S2x/LPC552x Flash API” for details.

9.1 How to read this chapter

This chapter applies to all LPC55S6x/LPC55S2x/LPC552x parts.

9.2 Features

- The internal flash stores the following information
  - The user application and the application data (in normal flash region).
  - The life-cycle related parameter update (in PFR region).
- Boot ROM API support for programming the flash region and the Protected Flash Region (PFR) region.

9.3 General description

The main purpose of these APIs is to simplify the use of flash driver APIs exported from the bootloader ROM

A set of parameters are required to ensure all APIs work properly.

This section describes how to use each flash driver API provided in the flash driver API tree.

9.3.1 ROM API structure

The ROM API table locates at address 0x130010f0. See Figure 46 for the ROM API layout.

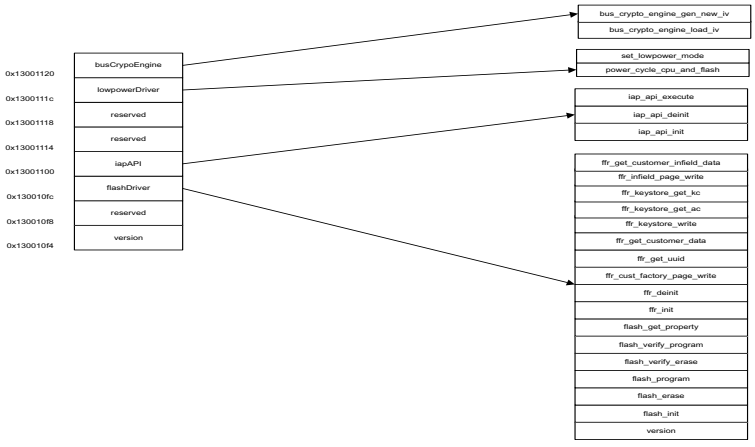


Fig 46. ROM API structure (for 1B silicon)

Note: It is suggested to use the fsl_iap.h, fsl_iap_ffr.h and fsl_iap.c in the MCUXpresso SDK release package directly. The API in these codes are applicable to all silicon revisions to date.

9.3.2 FLASH APIs

This section describes each function supported in the flash driver API.

The bootloader API prototypes are:

```
typedef struct FlashDriverInterface
{
    standard_version_t version; //!< flash driver API version number.
    // Flash driver
    status_t (*flash_init)(flash_config_t *config);
    status_t (*flash_erase)(flash_config_t *config, uint32_t start, uint32_t
     lengthInBytes, uint32_t key);
    status_t (*flash_program)(flash_config_t *config, uint32_t start, uint8_t *src,
     uint32_t lengthInBytes);
    status_t (*flash_verify_erase)(flash_config_t *config, uint32_t start, uint32_t
     lengthInBytes);
    status_t (*flash_verify_program)(flash_config_t *config,
```

```
                                    uint32_t start,
                                    uint32_t lengthInBytes,
                                    const uint8_t *expectedData,
                                    uint32_t *failedAddress,
                                    uint32_t *failedData);
    status_t (*flash_get_property)(flash_config_t *config, flash_property_tag_t
     whichProperty, uint32_t *value);
     uint32_t reserved[3]; /* Not for revision A0 */

    // Flash PFR driver
    status_t (*ffr_init)(flash_config_t *config);
    status_t (*ffr_deinit)(flash_config_t *config);
    status_t (*ffr_cust_factory_page_write)(flash_config_t *config, uint8_t*
     page_data, bool seal_part);
    status_t (*ffr_get_uuid)(flash_config_t *config, uint8_t* uuid);
   status_t (*ffr_get_customer_data)(flash_config_t *config, uint8_t* pData, uint32_t
     offset, uint32_t len);
    status_t (*ffr_keystore_write)(flash_config_t *config, ffr_key_store_t*
     pKeyStore);
   status_t (*ffr_keystore_get_ac)(flash_config_t *config, uint8_t* pActivationCode);
    status_t (*ffr_keystore_get_kc)(flash_config_t *config, uint8_t* pKeyCode,
     ffr_key_type_t keyIndex);
    status_t (*ffr_infield_page_write)(flash_config_t *config, uint8_t* page_data,
     uint32_t valid_len);
    status_t (*ffr_get_customer_infield_data)(flash_config_t *config, uint8_t* pData,
     uint32_t offset, uint32_t len);
} flash_driver_interface_t;
```

The user application can get the device revision through the following function:

```
static inline uint32_t Chip_GetVersion()
{
    uint32_t deviceRevision;

    deviceRevision = SYSCON->DIEID & SYSCON_DIEID_REV_ID_MASK;

    if (0 == deviceRevision) /* A0 device revision is 0 */
    {
        return 0x0;
    }
    else if (1 == deviceRevision) /* A1 device revision is 1 */
    {
        return 0x1;
    }
    else
    {
        return 0xFF;
    }
}
```

The flash_config_t is defined here:

```
/*! @brief Flash driver state information.
 *
 * An instance of this structure is allocated by the user of the flash driver and
 * passed into each of the driver APIs.
 */
typedef struct _flash_config
{
    uint32_t PFlashBlockBase;               /*!< A base address of the first PFlash
     block */
    uint32_t PFlashTotalSize;              /*!< The size of the combined PFlash block.
     */
    uint32_t PFlashBlockCount;              /*!< A number of PFlash blocks. */
    uint32_t PFlashPageSize;               /*!< The size in bytes of a page of PFlash.
     */
    uint32_t PFlashSectorSize;              /*!< The size in bytes of a sector of
     PFlash. */
    flash_ffr_config_t ffrConfig;
    flash_mode_config_t modeConfig;
} flash_config_t;
```

The flash_mode_config_t is defined here:

```
typedef struct _flash_mode_config
{
    uint32_t sysFreqInMHz;
    // ReadSingleWord parameter
    struct {
        uint8_t readWithEccOff : 1;
        uint8_t readMarginLevel : 2;
        uint8_t readDmaccWord : 1;
        uint8_t reserved0 : 4;
        uint8_t reserved1[3];
    } readSingleWord;
    // SetWriteMode parameter
    struct {
        uint8_t programRampControl;
        uint8_t eraseRampControl;
        uint8_t reserved[2];
    } setWriteMode;
    // SetReadMode parameter
    struct {
        uint16_t readInterfaceTimingTrim;
        uint16_t readControllerTimingTrim;
        uint8_t readWaitStates;
        uint8_t reserved[3];
    } setReadMode;
} flash_mode_config_t;
```

The flash_ffr_config_t is defined as below:

```
/*! @brief Flash controller paramter config. */
typedef struct _flash_ffr_config
```

```
{
    uint32_t ffrBlockBase;
    uint32_t ffrTotalSize;
    uint32_t ffrPageSize;
    uint32_t cfpaPageVersion;
    uint32_t cfpaPageOffset;
} flash_ffr_config_t;
```

9.3.2.1 flash_init

This API is used for initializing the flash controller and the flash_config context. It must be called before calling other flash APIs.

Prototype

```
status_t Flash_Init(flash_config_t *config);
```

Table 254. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.

Example:

```
#define ROM_API_TREE ((*uint32_t)0x130010f0)
#define FLASH_API_TREE ((flash_driver_interface_t*) ROM_API_TREE[3])
flash_config_t flashConfig;
status = FLASH_API_TREE->flash_init(&flashConfig);
```

See the possible status code in Section 9.3.3 “PFR APIs”

9.3.2.2 flash-erase

This API is used for erasing specified flash range.

Prototype

```
status_t FLASH_Erase(flash_config_t *config, uint32_t start, uint32_t lengthInBytes,
     uint32_t key);
```

Table 255. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
Start	The start address of the required flash memory to be erased. The start address must be page-aligned (that is, a multiple of 512) and the erase is page aligned same as the program.
lengthInBytes	The length, given in bytes (not words or long words) to be erased. Must be page-aligned.
Key	Key is used to validate erase operation. Must be set to "kFLASH_ApiEraseKey"

Example:

```
#define ERASE_KEY 0x6b65666b

status = FLASH_API_TREE->flash_erase(&flashConfig, 0x0, 0x4000, ERASE_KEY);
```

See the possible status code in Section 9.3.3 “PFR APIs”

9.3.2.3 flash_program

This API is used for programming data into specified flash region, the required *start* and the *lengthInBytes* must be page size aligned.

Prototype

```
status_t FLASH_Program(flash_config_t *config, uint32_t start, uint32_t *src, uint32_t
     lengthInBytes);
```

Table 256. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
Start	The start address of the required flash memory to be erased. The start address must be 512bytes-aligned.
src	Pointer to the source buffer of data that is to be programmed into flash.
lengthInBytes	The length in bytes (not words or long words) to be erased; the length must also be 512bytes-aligned.

Example:

```
uint8_t programBuffer[512];
for (uint32_t i=0; i<sizeof(programBuffer); i++)
{
     programBuffer[i] = (uint8_t)(i & 0xFF);
}
status = FLASH_API_TREE->flash_program(&flashConfig, 0x0, programBuffer,
                                       sizeof(programBuffer));
```

9.3.2.4 flash_verify_erase

This API is used to verify the erasure of the desired flash area.

This function checks the appropriate number of flash sectors based on the desired start address and length, to see if the flash has been erased.

Prototype

```
status_t FLASH_VerifyErase(flash_config_t *config, uint32_t start, uint32_t
     lengthInBytes);
```

Table 257. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
Start	The start address of the required flash memory to be verified. Must be page-aligned.
lengthInBytes	The length, given in bytes (not words or long words) to be verified. Must be page-aligned.

Example:

```
uint32_t propertyValue;

status = FLASH_API_TREE->flash_verify_erase(&flashConfig, 0x0, 0x4000);
```

9.3.2.5 flash_verify_program

This API is used to verify the data programmed in the flash memory and compares it with expected data for a given flash area (as determined by the start address and length).

Prototype

```
status_t FLASH_VerifyProgram(flash_config_t *config,

uint32_t start,

uint32_t lengthInBytes,

const uint32_t *expectedData,

uint32_t *failedAddress,

uint32_t *failedData);
```

Table 258. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
Start	The start address of the required flash memory to be verified.
lengthInBytes	The length, given in bytes (not words or long words) to be verified.
ExpectedData	Pointer to the expected data that is to be verified against.
FailedAddress	Pointer to returned failing address.
FailedData	Pointer to return failing data.

Example:

```
status = FLASH_API_TREE->flash_verify_program(&flashConfig, 0x0,
    0x4000, programBuffer, NULL, NULL);
```

9.3.2.6 flash_get_property

This API returns the required flash property, which includes base address, p size, and other options.

See Table 259 for supported properties.

Table 259. Parameters

Parameter	Value	Description
config		Pointer to flash_config_t data structure in memory to store driver runtime state.
whichProperty		The required property from the list of properties.
Property definition		
kFLASH_PropertyPflashTotalSize	0x01	Pflash total size property.
kFLASH_PropertyPflashBlockSize	0x02	Pflash Block size property.
kFLASH_PropertyPflashBlockCount	0x03	Pflash Block Count size property.
kFLASH_PropertyPflashBlockBaseAddr	0x04	flash block base address
kFLASH_PropertyPflashPageSize	0x30	Pflash page size property

Table 259. Parameters

Parameter	Value	Description
kFLASH_PropertyFfrTotalSize	0x41	PFR total size property
kFLASH_PropertyFfrBlockBaseAddr	0x42	PFR block base address property
kFLASH_PropertyFfrPageSize	0x43	PFR page size property

Example:

```
uint32_t propertyValue;

status = FLASH_API_TREE->flash_get_property(&flashConfig, 0x1,
    &propertyValue);
```

9.3.2.7 The flash driver status code

Table 260. Flash driver status code

Status	Code	Description
kStatus_FLASH_Success	0	The flash operation is successful.
kStatus_FLASH_InvalidArgument	4	Invalid argument detected during executing a FLASH API.
kStatus_FLASH_SizeError	100	Invalid size detected during executing a FLASH API.
kStatus_FLASH_AlignmentError	101	Alignment error detected during executing a FLASH API.
kStatus_FLASH_AddressError	102	Address error detected during executing a FLASH API.
kStatus_FLASH_AccessError	103	Access error detected during executing a FLASH API.
kStatus_FLASH_CommandFailure	105	Command failure detected during executing a FLASH API.
kStatus_FLASH_UnknownProperty	106	Unknown property for flash_get_property API.
kStatus_FLASH_EraseKeyError	107	Incorrect EraseKey for flash_erase API.
kStatus_FLASH_CommandNotSupported	111	An unsupported command is detected during executing a FLASH API.
kStatus_FLASH_EccError	116	ECC error detected during executing a FLASH API.
kStatus_FLASH_CompareError	117	Compare error detected during executing flash_erase_verify or flash_program_verify API.
kStatus_FLASH_RegulationLoss	118	Regulation loss detected during executing a FLASH API.
kStatus_FLASH_InvalidWaitStateCycles	119	The wait state cycle set to r/w mode is invalid.
kStatus_FLASH_OutOfDateCfpaPage	132	CFPA page version is out of date.
kStatus_FLASH_BlankIfrPageData	133	Blank page cannnot be read.
kStatus_FLASH_EncryptedRegionsEraseNotDoneAtOnce	134	Encrypted flash subregions are not erased at once.
kStatus_FLASH_ProgramVerificationNotAllowed	135	Program verification is not allowed when the encryption is enabled.
kStatus_FLASH_HashCheckError	136	Hash check of page data is failed.
kStatus_FLASH_SealedFfrRegion	137	The FFR region is sealed.
kStatus_FLASH_FfrRegionWriteBroken	138	The FFR Spec region is not allowed to be written discontinuously.
kStatus_FLASH_NmpaAccessNotAllowed	139	The NMPA region is not allowed to be read/written/erased.
kStatus_FLASH_CmpaCfgDirectEraseNotAllowed	140	The CMPA Cfg region is not allowed to be erased directly.
kStatus_FLASH_FfrBankIsLocked	141	The FFR bank region is locked.

9.3.3 PFR APIs

This section describes each function supported in the PFR driver API.

Note: FFR write and flash erase commands to FFR regions are inhibited if a region has a SHA256 hash digest field programmed into the last 32 bytes (256 bits).

9.3.3.1 ffr_init

This API is used for initializing the PFR controller and the flash_ffr_config context, it must be called before calling other PFR APIs.

Note: flash_init (see: Section 9.3.2.1 “flash_init”) must be called before calling the ffr_init API.

Prototype

```
status_t ffr_init (flash_config_t *config);
```

Table 261. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.

Example:

```
status = FLASH_API_TREE->ffr_init(&flashConfig);
```

9.3.3.2 ffr_deinit

This API is used to enable firewall for all flash banks, include enable flash protection for three IFR banks and disable write access to FLASHBENKENABLE register. ffr_deinit unconditionally locks writing to CFPA, CMPA, and NMPA flash areas. Writes will subsequently be inhibited unless a power-on reset (POR) or brown-out detect (BOD) reset occurs.

Prototype

```
status_t ffr_deinit (flash_config_t *config);
```

Table 262. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.

Example:

```
Status = FLASH_API_TREE->ffr_deinit(&flashConfig);
```

9.3.3.3 ffr_cust_factory_page_write

This API is used to access CMPA pages. it will erase *Customer Factory Page* and program the page with passed data.

Prototype

```
status_t ffr_cust_factory_page_write (flash_config_t *config, uint8_t* page_data, bool
     seal_part);
```

Table 263. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
Page date	Pointer to value address that will be written to the destination address.
seal_part	If seal_part is TRUE then the routine will compute SHA256 hash of the page contents and then programs the pages. 1. During development customer code uses this API with 'seal_part' set to FALSE. 2. During manufacturing this parameter should be set to TRUE to seal the part from additional modifications. Cleanup temp page buffer.

Example:

```
uint32_t page_data[ffr_page_size] = {0, 2, 3, 4};
status = FLASH_API_TREE-> ffr_cust_factory_page_write(&flashConfig, (uint8_t*)&
     page_data, false);
```

9.3.3.4 ffr_get_uuid

This API is used to get the UUID from the NXP programmed Page Area.

Prototype

```
status_t ffr_get_uuid (flash_config_t *config, uint8_t* uuid)
```

Table 264. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
uuid	Pointer to value address, the value is read back from the nmpa configuration uuid

Example:

```
uint32_t uuid[4];
Status = FLASH_API_TREE->ffr_get_uuid(&flashConfig, (uint8_t*) uuid);
```

9.3.3.5 ffr_get_customer_data

This API is used to read data stored in *Customer Factory Page*.

Prototype

```
status_t ffr_get_customer_data (flash_config_t *config, uint8_t* pData, uint32_t
     offset, uint32_t len);
```

Table 265. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pDate	Point to the destination buffer of date that stores data read from Customer Factory Page.
offset	Point to the source address of data is to be read.
len	The length in bytes to be read back.

Example:

```
uint32_t pData [4];

status = FLASH_API_TREE->ffr_get_customer_data(config, (uint8_t*)pData, 0,
     sizeof(pData));
```

9.3.3.6 ffr_keystore_write

This API is used to writes the three pages allocated for key store data.

Prototype

```
status_t ffr_keystore_write (flash_config_t * flashConfig, ffr_key_store_t*
     pKeyStore);
```

Table 266. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pKeyStore	Pointer to ffr_key_store_t date structure, to store 3 pages allocated by the key.

Example:

```
ffr_key_store_t pKeyStore;
Status = FLASH_API_TREE->ffr_keystore_write (&flashConfig, &pKeyStore);
```

9.3.3.7 ffr_keystore_get_ac

This API is used to get the Activation code from the Key Store Area. Calling code should pass buffer pointer which can hold activation code (1192 bytes).

Remark: Check if the flash aperture is small or regular and read the data appropriately.

Prototype

```
status_t ffr_keystore_get_ac (flash_config_t *config, uint8_t* pActivationCode)
```

Table 267. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pActivationCode	Point to the destination buffer of data that stores the Activation Code read from Key Store Area. the buffer must be able to hold a minimum of 1192 bytes.

Example:

```
#define BUF_LEN 1192
Uint32_t pActivationCode[BUF_LEN / sizeof(uint32_t)];
Status_t status = FLASH_API_TREE-> ffr_keystore_get_ac(&flashconfig,
     (uint8_t)pActivationCode);
```

9.3.3.8 ffr_keystore_get_kc

This API is used to get key codes from the Key Store Area. The calling code should pass buffer pointer which can hold key code 52 bytes.

Remark: Check if flash aperture is small or regular and read the data appropriately.

Prototype

```
status_t ffr_keystore_get_kc (flash_config_t *config, uint8_t* pKeyCode,
     ffr_key_type_t keyIndex);
```

Table 268. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pKeyCode	Point to the destination buffer of data that stores the Key Code read from the Key Area, the destination buffer must be able to hold a minimum of 52 bytes.
keyIndex	Declare an enumeration variable of type ffr_bank_type_t

Example:

```
#define BUF_LEN 1192
ffr_key_type_t keyIndex;
status_t status = FLASH_API_TREE-> ffr_keystore_get_kc(&flashconfig, pKeyCode,
     keyIndex);
```

9.3.3.9 ffr_infield_page_write

This API is used to program the in-field page.

Prototype

```
status_t ffr_infield_page_write (flash_config_t *config, uint8_t* page_data, uint32_t
     valid_len);
```

Table 269. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
page_data	Pointer to the source buffer of data that is to be programmed into in-field page.
valid_len	The length in bytes to be programmed, the length must equal the page size.

Example:

```
uint32_t page_data [128];
status = FLASH_API_TREE-> ffr_infield_page_write(&flashConfig, (uint8_t *)page_data,
     sizeof(page_date));
```

9.3.3.10 ffr_get_customer_infield_data

This API is used to Read data stored in *Customer In-field Page.*

Prototype

```
status_t ffr_get_customer_infield_data (flash_config_t *flashConfig, uint8_t* pData,
     uint32_t offset, uint32_t len);
```

Table 270. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.

Table 270. Parameters

Parameter	Description
pData	Point to the destination buffer of data that stores data read from Customer In-field page.
offset	Point to the offset data to read.
len	Point to the length of data to read.

Example:

```
Uint32_t pDate [4];
Status = FLASH_API_TREE-> ffr_get_customer_infield_data(&flashConfig, (uint8_t *)
     pData, offset, len );
```

Note: Please refer to the attached ROM_API document.

9.3.4 runBootloader API

The ROM bootloader provides an API for the user application to enter the ISP mode based on the designated ISP interface mode.

Prototype

```
void (*runBootloader)(void *arg);
```

Table 271. API prototype fields

Field	Offset	Description
Tag	[31:24]	Fixed value: 0xEB (Enter Boot)
Boot Mode	[23:20]	0 - Enter passive mode
		1 - Enter ISP mode
ISP Interface	[19:16]	0 - Auto detection
		1 - USB-HID
		2 - UART
		3 - SPI
		4 - I2C
Reserved	[15:04]	
Image Index	[03:00]	Used for Boot Mode 0

Example: In the application image boot process, regardless of whether the ISP pin is connected to the high level, the device will directly enter the ISP mode through the UART interface according to the parameter ispInterface in arg.

```
#define BOOTLOADER_TREE_LOCATION (0x1301fe00)

bootloader_tree_t *romApiTree = (bootloader_tree_t *)BOOTLOADER_TREE_LOCATION;

uint32_t arg = 0xEB120000;   //0xEB: represents Enter Boot; 0x12: represents enter ISP

mode by UART only

void runBootloader(void *arg)

{
```

```
romApiTree-> runBootloader(&arg);
}
```

After the application image, which calls the above runBootloader API, has booted successfully, the device will only allow the UART interface to be connected to transfer the data with the host.

10.1 How to read this chapter

The Protected Flash Region is included on all LPC55S6x/LPC55S2x/LPC552x devices

10.2 General description

The LPC55S6x/LPC55S2x/LPC552x family consists of an internal protected flash region (PFR) which can be accessed using ROM APIs. During boot time, ROM locks the PFR.

There are three regions defined within the protected flash region.

Table ”PFR memory table” shows the CFPA scratch, ping, and pong pages addresses.

Note: For bit field descriptions, see the attached Protected Flash Region.xls for further details.

Table 272. PFR memory table

Page	Address
Customer In-Field Programmable Area (CFPA)	0x9DE00 - 0x9E3FF
Customer Manufacturing / Factory Programmable Area (CMPA)	0x9E400 - 0x9EBFF
NXP Manufacturing Programmed Area (NMPA)	0x9EC00 - 0x9FDFF

10.2.1 Customer Manufacturing Programmable Area (CMPA)

In this region, the user can set the following features:

- Boot Configuration: User can define the boot speed where Core clock can be set to 48MHz FRO or 96MHz FRO. Default ISP mode (Auto ISP, USB (0 or 1) HID ISP, UART ISP, SPI Slave ISP, I2C slave ISP, and Disable ISP). Boot Failure Indication using GPIO port pin (default is PIO0_0).
- USB Product ID and USB Vendor ID
- Security Policy for Debug access control:
  CC_SOCU_PIN

  With TZ-M, the part can be sold by level 1 customers (secure code developer) to level-2 customers who develops non-secure code only. In this scenario, for ease of development, Level-I customer releases the part to always allow non-secure debug. To allow level-2 customers to further seal the part, the DCFG_CC_SOCU_NS word is used. ROM will use this word to further restrict the debug access.

  CC_SOCU_DFLT

  With TZ-M, the part can be sold by level 1 customers (secure code developer) to level-2 customers who develops non-secure code only. In this scenario, or easy of development, Level-I customer releases the part to always allow non-secure debug. To allow level-2 customers to further seal the part , DCFG_CC_SOCU_NS is used. ROM will use this word to further restrict the debug access.

DAP_VENDOR_USAGE_FIXED

Vendor Usage field is used by vendor and it's interpretation is application specific. It can be used during debug authentication and the debugger supplies an authorized debug credential that has a value that matches the attribute in the device configuration. Use-cases for this authorization constraint include different product families administered under the same RoT, regions/jurisdictions/domains, so that different debug credentials are required for debugging on different domains and product versions, so that newer debug credentials are required for debugging newer devices. If the vendor sub-divides the Vendor Usage (VU) attribute bit-wise, a combination of these different interpretations could be supported.

- Secure boot flags (SECURE_BOOT_CFG) allows user to perform plain image boot with or without CRC, RSA signed image boot, enable or disable PUF enrollment, PUF key code generation, boot to secure mode or non-secure mode, and use RSA4096 keys.
- Prince region configurations
- Root of Trust Keys Table hash
- Key Store Area: Device specific PUF activation code, Secure Binary Key Encryption Key (SBKEK key code) used for SB2 firmware update image decryption, User Key Encryption Key (KEK key code) used for user as pre-shared master key (256-bit symmetric key), and Prince region 0 to 2 keys (128-bit symmetric key) used to encrypt/decrypt data internal flash memory when PRINCE is enabled for given memory region.

10.2.2 Customer Field Programmable Area (CFPA)

In this region, the user can set the following features:

- Three Monotonic counters where its version must be higher (increment only) or equal: Secure firmware version used during SB2 file loading, Non-secure firmware version used during SB2 file loading, and Image key ID revocation ID version which is checked during image authentication process.
- RoT (RKTH) Key revocation of four RoT keys
- Prince region IV codes used to configure IV for PRINCE regions.

Two pages in independent protected area are provided and used for CFPA. Page with higher version number is picked as active page by ROM. Pages in the CFPA are updated using ROM API and following steps:

Application code uses API to update the scratch page which remains outside the protected region.

- Core is reset to make the page effective.

  On subsequent boot, ROM checks if the scratch page is valid and has higher version.

  ROM erases the oldest of the two protected pages (ping pong pages).

  Copies the scratch page contents to the erased area.

10.2.3 NXP Programmed Area

In this region, the user can find the UUID (universal unique identification ID) which can be used to uniquely identify a device.

10.2.4 Region SHA256 Hash Digest

Each region has a SHA256 hash digest field in the last 32 bytes (256 bits). The hash digest is used during the deployment life-cycle state to cross check the integrity of the region. Programming a region's respective hash digest blocks, erase/writes commands to that region by the ROM or by flash API commands executed by the run-time code.

10.3 LPC55S69 Customer Development Lifecycle state

Customers will initially use the LPC55S6x/LPC55S2x/LPC552x device in the “NXP closed” lifecycle state. Depending on application security targets, customers can ship the LPC55S6x/LPC55S2x/LPC552x in this “NXP closed” state, or, move it to the “OEM closed” state, were the user programmable domain will require an authenticated image to boot.

The “Returned” state (a.k.a. FA mode) is used for retiring the LPC55S6x/LPC55S2x/LPC552x, not before erasing various secrets that were provisioned, by NXP and Tier1/2/OEM. This state will allow running FA testing on customer returns. The PUF output is completely blocked and the part doesn't boot any images. It only provides debug access.

The details on each of the states (e.g. PFR fields that needs to be programmed, impact on each subsystem) and transitioning between states (PFR fields that need to be programmed) are provided in the table below:

Table 273. Lifecycle state descriptions

Life cycle state	Descriptions	PFR Fields to provision	Transitions		
			Fields to Write	Next State	Notes
NXP closed	Initial customer state: Debug ports opened by ROM. But debug access is disabled during ROM execution. If debug authentication fields are programmed then they are used to determine debug access.	CMPA fields are programable: Secure boot: • ROTKH[255:0] • BOOT_CFG • SECURE_BOOT_CFG PRINCE: • PRINCE_BASE_ADDR • PRINCE_SR_0 • PRINCE_SR_1 • PRINCE_SR_2 Debug Authentication: • CC_SOCU_PIN • CC_SOCU_DFLT • VENDOR_USAGE PUF Keys store: • Activation code • PRINCE0 key • PRINCE1 key • PRINCE2 key • SBKEK • User KEK Optional: • USB_ID • Customer defined fields • CFPA fields	CMPA_DIGEST (SHA256 hash of CMPA pages) are left blank.	Tier 1 Dev	-

Table 273. Lifecycle state descriptions

Life cycle state	Descriptions	PFR Fields to provision	Transitions		
			Fields to Write	Next State	Notes
Tier 1 Dev.	Tier 1 Dev. Tier 1 software development state: • Debug ports opened by ROM. But debug access is disabled during ROM execution. • If debug authentication fields (CC_SOCU_xxx) are programmed then they are used to determine debug access. • Once CMPA hash is programmed then fields in CMPA region are not changeable. • SECURE_BOOT_CFG field determines whether secure boot flow is enabled or not. • If secure boot is enabled or debug authentication fields (CC_SOCU_xxx) are not in the default state, then limited ISP commands are allowed. Allowed command set can be retrieved by “blhost -p COMx/-u <VID,PID> --get-property 7”.	CMPA fields are programable: Secure boot: • ROTKH[255:0] • BOOT_CFG • SECURE_BOOT_CFG PRINCE: • PRINCE_BASE_ADDR • PRINCE_SR_0 • PRINCE_SR_1 • PRINCE_SR_2 Debug Authentication: • CC_SOCU_PIN • CC_SOCU_DFLT • VENDOR_USAGE PUF Keys store: • Activation code • PRINCE0 key • PRINCE1 key • PRINCE2 key • SBKEK • User KEK Optional: • USB_ID Customer defined fields	• CMPA_DIGEST is programmed. • But Debug Authentication configuration are set to make non-secure development open for Tier2 development. • CC_SOCU_PIN • CC_SOCU_DFLT • VENDOR_USAGE	Tier 2 Dev	-
		Same as above	CMPA_DIGEST is programmed. Debug Authentication configuration are set to closed state (ie., disabled permanently or enable only after authentication). CFPA_DIGEST	OEM closed	See the return sequence notes, in the "Returned" state.
		Same as above	• CFPA field ENABLE_FA_MODE is set.	Returned	-

Table 273. Lifecycle state descriptions

Life cycle state	Descriptions	PFR Fields to provision	Transitions		
			Fields to Write	Next State	Notes
Tier 2 Dev	Tier 2 software development state: • Secure Debug ports closed (unless authenticated) or always closed • Non-secure Debug enabled • TZ-M is enabled by primary image. CMPA cannot be written	CFPA fields are programmable through secure API calls exposed by customer code. • CC_SOCU_PIN_NS and CC_SOCU_DFLT_NS fields in CFPA are used to close the part further.	CC_SOCU_PIN_NS and CC_SOCU_DFLT_NS fields in CFPA	OEM closed	
			ENABLE_FA_MODE	Returned	See the return sequence notes, in the "Returned" state.
OEM Closed	In-field Application State: • Debug ports closed (unless authenticated) or always closed • TZ-M optional • CMPA cannot be written CFPA can only be written scratch page mechanism		CFPA fields are programmable through secure API as needed through the product life: CTRK_REVOKE[3:0] IMG_KEY_REVOKE[15:0] SS_VER_CNT[63:0] NS_VER_CNT[255:0] VENDOR_USAGE (debug key revoke)	Returned	See the return sequence notes, in the "Returned" state.
	Returned (CQI): • ROM checks key store is empty and blocks PUF key unwrapping functionality before enabling Test / Debug ports open. • ROM doesn't boot images in flash but stays in while(1) loop. • Part can only be used to run test patterns through SWD or load code to RAM and run using SWD interface. • ISP command interface is also disabled. • CMPA & CFPA cannot be written	N/A	The following sequence of operations is required to transition a part in the "Returned state": Erase flash (mass erase, except PFR) Erase System SRAM Erase PUF key store Set ENABLE_FA_MODE field Trigger Reset		

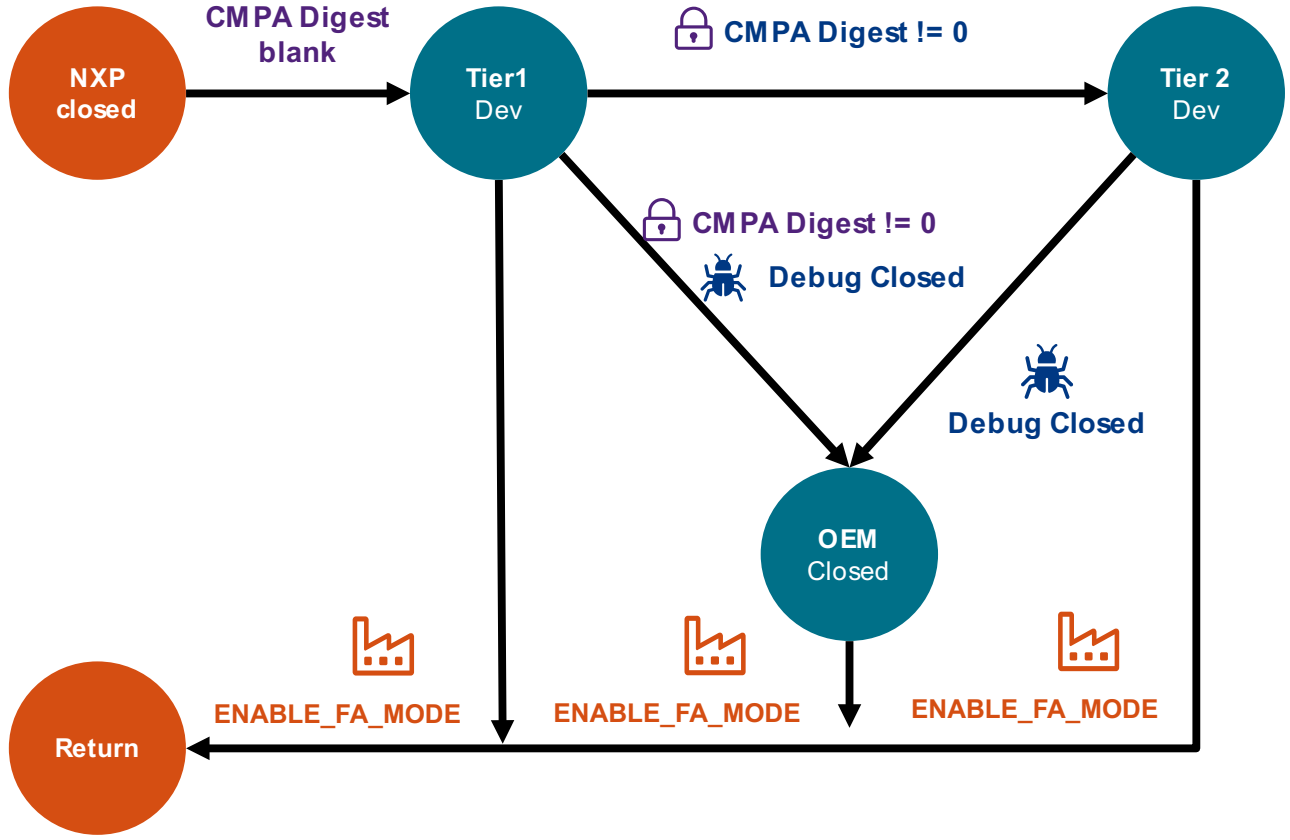


Fig 47. Customer Development Lifecycle State

10.4 Firewall PFR pages

Firewall PFR pages protect from modifications (erase/write) with a lock mechanism or firewall depending on the current lifecycle state.

Table 274. Lifecycle Updates state

Page	Address
NXP Closed	NMPA region is locked. CMPA and CFPA open for modifications.
Tier 1 Development	NMPA region is locked. CMPA and CFPA open for modifications.
Tier 2 Development	All regions are locked. CFPA updates are available through the scratchpad mechanism.
OEM Closed	All regions are locked. CFPA updates are available through the scratchpad mechanism.
Return	All regions are locked. CFPA scratchpad disable.

Note: CFPA scratchpad updates need to issue a PoR or SYSCON_SWRESET when the PFR lifecycle is in Tier 2 Development or OEM Closed.

11.1 How to read this chapter

The analog controller is available on all LPC55S6x/LPC55S2x/LPC552x devices.

11.2 Features

- Internal Free Running Oscillator (FRO). This oscillator provides a selectable 96 MHz output, and a 12 MHz output (divided down from the selected higher frequency) that can be used as a system clock.
- 32 kHz internal FRO.
- High-accuracy frequency measurement function for on-chip and off-chip clocks.
- 32 MHz Crystal oscillator module control and status.
- All Brown out Detectors (BoD) and DCDC converter interrupts generation control and status.
- Ring oscillators (True Random Number Generator Clock sources) functions control.
- All Crystal oscillators (both 32 kHz and high-speed 16 MHz to 32 MHz) capacitive banks calibration functions control.
- Some USB high-speed physical interface parameters control.

11.3 Basic configuration

- Set the ANALOG_CTRL bit in the AHBCLKCTRL2 register Section 4.5.19 “AHB clock control 2” to enable the clock to the analog controller module. **NOTE:** The clock to analog controller module is enabled during boot time by the boot loader and should always remain enabled.
- Target and reference clocks for the frequency measurement function are selected in the input multiplexer block. See Section 18.6.9 “Frequency measure function reference clock select register” and Section 18.6.10 “Frequency measure function target clock select register”.
- The 32 kHz Free Running Oscillator is automatically enabled when: the RTC_OSP_PD bit in RTC control register = 0 and the SEL bit in RTCOSC32K register = 0 or when the OSC32KPD bit in the OSTIMER register = 0 and the SEL bit in the RTCOSC32K = 0 or when the PDEN_FRO32K bit in PDRUNCFG0 = 0.

11.3.1 Measure the frequency of a clock signal

The frequency of any on-chip or off-chip clock signal can be measured accurately with a selectable reference clock. For example, the frequency measurement function can be used to accurately determine the frequency of the 32 kHz Free Running Oscillator (FRO32kHz).

The clock frequency to be measured and the reference clock are selected in the input mux block. See Section 18.6.9 “Frequency measure function reference clock select register” and Section 18.6.10 “Frequency measure function target clock select register”. Details on the accuracy and measurement process are described in Section 11.6.1 “Frequency measure function”. To start a frequency measurement cycle and read the result, see Table 278.

11.4 Pin description

The analog controller has no configurable pins.

11.5 Register description

Table 275. Register overview: ANACTRL (base address = 0x50013000)

Name	Access	Offset	Description	Reset value	Section
ANALOG_CTRL_CFG	RW	0x0	Various Analog blocks configuration (like FRO 192MHz trimmings source...).	0x0000 0000	11.5.1
ANALOG_CTRL_STATUS	R	0x4	Analog macroblock identity registers, flash status registers.	0x5000 0000	11.5.2
FREQ_ME_CTRL	RW	0xC	Frequency measure function control register.	0x0000 0000	11.5.3
FRO192M_CTRL	RW	0x10	FRO 192 MHz control register.	0x0080 D01A	11.5.4
FRO192M_STATUS	R	0x14	FRO 192 MHz status register.	0x0000 0003	11.5.5
ADC_CTRL	RW	0x18	General Purpose ADC VBAT Divider branch control.	0x0000 0000	11.5.6
XO32M_CTRL	RW	0x20	32 MHz Crystal oscillator control register.	0x0021 428A	11.5.7
XO32M_STATUS	R	0x24	32 MHz Crystal oscillator status register.	0x0000 0000	11.5.8
BOD_DCDC_INT_CTRL	RW	0x30	BoDs & DCDC interrupts generation control register.	0x0000 0000	11.5.9
BOD_DCDC_INT_STATUS	R	0x34	BoDs & DCDC interrupts status register.	0x0000 0105	11.5.10
LDO_XO32M	RW	0xB0	High Speed Crystal Oscillator (16 MHz - 25 MHz) Voltage Source Supply Control register.	0x0000 03A0	11.5.11
AUX_BIAS	RW	0xB4	Control output of 1V reference voltage.	0x0007 03A0	11.5.12
USBHS_PHY_TRIM	RW	0x104	USB High Speed Phy Trim values	0x0000 0000	11.5.13

11.5.1 Various Analog blocks configuration (like FRO 192MHz trimmings source …) (ANALOG_CTRL_CFG)

Disclaimer: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

The Analog Control Configuration register gathers miscellaneous general configurations related to various analog modules.

Table 276. (ANALOG_CTRL_CFG, offset = 0x0)

Bit	Symbol	Access Value	Value	Description	Reset value
0	FRO192M_TRIM_SRC	RW		FRO192M trimming and 'Enable' source.	0x0
			0	FRO192M trimming and 'Enable' comes from eFUSE.	
			1	FRO192M trimming and 'Enable' comes from FRO192M_CTRL registers.	
31:1	-	W	-	Reserved. Read value is undefined, only zero should be written.	-

11.5.2 Analog control status register

The analog control status register can be used to determine the status of the flash.

Table 277. (ANALOG_CTRL_STATUS, offset = 0x4)

Bit	Symbol	Value	Description	Reset value
11:0	-		Reserved. Read value is undefined, only zero should be written.	-
12	FLASH_PWRDWN		Flash power-down status.	0x0
		0	Flash is not in power-down mode.	
		1	Flash is in power-down mode.	
13	FLASH_INIT_ERROR		Flash initialization error status.	0x0
		0	No error.	
		1	At least one error occurs during the flash initialization.	
31:14	-		Reserved. Read value is undefined, only zero should be written.	-

11.5.3 Frequency measure function control register

The Frequency Measure function control register starts the frequency measurement function and stores the result in the CAPVAL field. The target frequency can be calculated as follows with the frequencies given in MHz:

Ftarget = (CAPVAL * Freference) / ((1<<SCALE)-1)

Select the reference and target frequencies using the FREQMEAS_REF and FREQMEAS_TARGET before starting a frequency measurement by setting the PROG bit in FREQ_ME_CTRL.

Table 278. (FREQ_ME_CTRL, offset = 0xC)

Bit	Symbol	Value	Description	Reset value
30:0	CAPVAL_SCALE		CAPVAL = FREQMECTRL[30:0] (Read-only). Stores the capture result which is used to calculate the frequency of the target clock. SCALE = FREQ_ME_CTRL[4:0] (Write-only). Define the power of 2 count that the reference counter counts to during measurement. Note that the minimum count is 2 ie $2^2 = 4$.	0x0
31	PROG		Set this bit to 1 to initiate a frequency measurement cycle. Hardware clears this bit when the measurement cycle has completed and there is valid capture data in the CAPVAL field (bits 30:0).	0x0

Also see:

- Section 11.3.1 “Measure the frequency of a clock signal”
- Section 11.6.1 “Frequency measure function”
- Frequency reference clock select register (FREQMEAS_REF) - Section 11.6.5
- Frequency target clock select register (FREQMEAS_TARGET) - Section 11.6.6

11.5.4 FRO192M control register

The FRO192M control register is used to configure the on-chip high-speed Free Running Oscillator (FRO192 MHz).

Table 279. FRO 192M control register (FRO192M_CTRL, offset = 0x10)

Bit	Symbol	Value	Description	Reset value
13:0	-		Reserved. Read value is undefined, only zero should be written.	-
14	ENA_12MHZCLK		12 MHz clock control.	0x1
		0	12 MHz clock is disabled.	
		1	12 MHz clock is enabled.	
15	-		Reserved. Only 1 should be written. Writing zero prevents the Flash from working.	0x1
		0	Reserved. Specifying this value prevents the Flash from working.	
		-	-	
23:16	DAC_TRIM		Frequency trim. Boot code configures this to a device-specific factory trim value for the FRO. If USBCLKADJ = 1, this field is read-only and provides the value resulting from USB rate adjustment. See the USBMODCFG flag regarding reading this field. Application code may adjust this field when USBCLKADJ = 0. A single step of the DAC_TRIM is roughly equivalent to 0.1% of the selected FRO frequency.	0x80
24	USBCLKADJ		If this bit is set and the USB peripheral is enabled to full-speed device mode, the USB block will provide FRO clock adjustments to lock it to the host clock using the SOF packets.	0x0
25	USBMODCHG		If it reads as 1 when reading the DAC_TRIM field and USBCLKADJ=1, it should be re-read until it is 0.	0x0
29:26	-	-	Reserved. Read value is undefined, only zero should be written.	0x0
30	ENA_96MHZCLK		96 MHz clock control.	0x0
		0	96 MHz clock is disabled.	
		1	96 MHz clock is enabled.	
31	WRTRIM		This must be written to 1 to modify the BIAS_TRIM and TEMP_TRIM fields.[1]	0x0

[1] Disclaimer: The value written to this bit should not be changed. This bit is handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

Notes on using USBCLKADJ (automatic USB rate adjustment mode):

First, the bit field FRO192M_TRIM_SRC in register ANALOG_CTRL_CFG must set to ‘1’.

When turning on USBCLKADJ, the current DAC_TRIM value will be used as the starting value. From then on, the adjusted value will be used as long as enabled (whether USB is active or not).

If USBCLKADJ is turned off, the application may perform one of the following actions:

- Read the register to pick up the adjusted DAC_TRIM and then write back with the USBADJ cleared. The FRO will continue to use the adjusted value.
- If software saved the original factory trimmed value of DAC_TRIM, it can be written back as above.

11.5.5 FRO192M status register

High Speed Free Running Oscillator (FRO) Status Register.

Table 280. FRO 192M status register (FRO192M_STATUS, offset = 0x14)

Bit	Symbol	Value	Description	Reset value
0	CLK_VALID		Output clock valid signal.	0x1
		0	No output clock present (None of 12 MHz, or 96 MHz clock is available).	
		1	Clock is present (12 MHz, or 96 MHz can be the output if they are enabled respectively by FRO192M_CTRL.ENA_12MHZCLK/ENA_96MHZCLK).	
1	ATB_VCTRL		CCO threshold voltage detector output (signal vcco_ok).	0x0
31:2	-		Reserved. Read value is undefined, only zero should be written.	-

11.5.6 General Purpose ADC VBAT Divider branch control

Disclaimer: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

The following table provides the bit descriptions of General Purpose ADC VBAT Divider branch control.

Table 281. ADC VBAT Divider branch control (ADC_CTRL, offset = 0x18)

Bit	Symbol	Access Value	Value	Description	Reset value
0	VBATDIVENABLE	RW		Switch On/Off VBAT divider branch.	0x0
			0	VBAT divider branch is disabled.	
			1	VBAT divider branch is enabled.	
31-1	-	W		Reserved. Read value is undefined, only zero should be written.	-

11.5.7 32 MHz crystal oscillator control register

The following table provides the bit descriptions for the 32 MHz crystal oscillator control register.

Table 282. 32 MHz Crystal oscillator control register (XO32M_CTRL, offset = 0x20)

Bit	Symbol	Access	Value	Description	Reset value
0		WO		Reserved. Read value is undefined, only zero should be written.	-
3:1	GM	RW		Gm value for Xo.[1]	0x5
4	SLAVE	RW		Xo in slave mode.[1]	0x0
7:5	AMP	RW		Amplitude selection, Min amp: 001, Max amp: 110.[1]	0x4
14:8	OSC_CAP_IN	RW		Tune capa banks of High speed crystal Oscillator input pin.[1]	0x6
21:15	OSC_CAP_OUT	RW		Tune capa banks of High speed crystal Oscillator output pin.[1]	0x6
22	ACBUF_PASS_ENABLE	RW		Allows XO32M to be configured in bypass mode.	0x0
			0	XO bypass is disabled.	
			1	XO bypass is enabled.	
23	ENABLE_PLL_USB_OUT	RW		Enable XO 32 MHz output to USB HS PLL.	0x0
			0	XO 32 MHz output to USB HS PLL is disabled.	
			1	XO 32 MHz output to USB HS PLL is enabled.	

Table 282. 32 MHz Crystal oscillator control register (XO32M_CTRL, offset = 0x20) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
24	ENABLE_SYSTEM_CLK_OUT	RW		Enable XO 32 MHz output to CPU system, SCT, and CLKOUT	0x0
			0	XO 32 MHz output to CPU system is disabled.	
			1	XO 32 MHz output to CPU system is enabled.	
28:25	-	RW		Reserved.	0x0
31:29	-	WO		Reserved. Read value is undefined, only zero should be written.	-

[1] Disclaimer: The value written to this bit should not be changed. This bit is handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

11.5.8 32 MHz crystal oscillator status register

High Speed Crystal Oscillator (16 MHz - 32 MHz) - also referred as "XO 32 MHz" - output Control Register.

Table 283. 32 MHz Crystal oscillator status register (XO32M_STATUS, offset = 0x24)

Bit	Symbol	Access	Value	Description	Reset value
0	XO_READY	RO		Indicates XO 32 MHz out frequency stability.	0x0
			0	XO 32 MHz output frequency is not yet stable.	
			1	XO 32 MHz output frequency is stable.	
31:1	-	RO		Reserved. Read value is undefined.	-

11.5.9 Brown Out Detectors (BoDs) and DCDC interrupts generation control register

This register is used to manage interrupts from BoD VBAT, BoD CORE and DCDC.

Table 284. Brown Out Detectors (BoDs) and DCDC interrupts generation control register (BOD_DCDC_INT_CTRL, offset = 0x30)

Bit	Symbol	Access	Value	Description	Reset value
0	BODVBAT_INT_ENABLE	RW		BOD VBAT interrupt control.	0x0
			0	BOD VBAT interrupt is disabled.	
			1	BOD VBAT interrupt is enabled.	
1	BODVBAT_INT_CLEAR	RW		BOD VBAT interrupt clear.1: Clear the interrupt. Self-cleared bit.	0x0
2	BODCORE_INT_ENABLE	RW		BOD CORE interrupt control.	0x0
			0	BOD CORE interrupt is disabled.	
			1	BOD CORE interrupt is enabled.	
3	BODCORE_INT_CLEAR	RW		BOD CORE interrupt clear.1: Clear the interrupt. Self-cleared bit.	0x0

Table 284. Brown Out Detectors (BoDs) and DCDC interrupts generation control register (BOD_DCDC_INT_CTRL, offset = 0x30) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
4	DCDC_INT_ENABLE	RW		DCDC interrupt control.	0x0
			0	DCDC interrupt is disabled.	
			1	DCDC interrupt is enabled.	
5	DCDC_INT_CLEAR	RW		DCDC interrupt clear.1: Clear the interrupt. Self-cleared bit.	0x0
31:6	-	WO		Reserved. Read value is undefined, only zero should be written.	-

11.5.10 BOD_DCDC_INT status register

This register provides the output status and the interrupt status from BoD VBAT, BoD CORE and DCDC.

Table 285. BoDs and DCDC interrupts status register (BOD_DCDC_INT_STATUS, offset = 0x34)

Bit	Symbol	Access	Value	Description	Reset value
0	BODVBAT_STATUS	RO		BOD VBAT Interrupt status before Interrupt Enable.	0x1
			0	No interrupt pending.	
			1	Interrupt pending.	
1	BODVBAT_INT_STATUS	RO		BOD VBAT Interrupt status after Interrupt Enable.	0x0
			0	No interrupt pending.	
			1	Interrupt pending.	
2	BODVBAT_VAL	RO		Current value of BOD VBAT power status output.	0x1
			0	VBAT voltage level is below the threshold.	
			1	VBAT voltage level is above the threshold.	
3	BODCORE_STATUS	RO		BOD CORE Interrupt status before Interrupt Enable.	0x0
			0	No interrupt pending.	
			1	Interrupt pending.	
4	BODCORE_INT_STATUS	RO		BOD CORE Interrupt status after Interrupt Enable.	0x0
			0	No interrupt pending	
			1	Interrupt pending.	
5	BODCORE_VAL	RO		Current value of BOD CORE power status output.	0x0
			0	CORE voltage level is below the threshold.	
			1	CORE voltage level is above the threshold.	
6	DCDC_STATUS	RO		DCDC Interrupt status before Interrupt Enable.	0x0
			0	No interrupt pending.	
			1	Interrupt pending.	
7	DCDC_INT_STATUS	RO		DCDC Interrupt status after Interrupt Enable.	0x0
			0	No interrupt pending.	
			1	Interrupt pending.	

Table 285. BoDs and DCDC interrupts status register (BOD_DCDC_INT_STATUS, offset = 0x34) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
8	DCDC_VAL	RO		Current value of DCDC power status output.	0x1
			0	DCDC output Voltage is below the targeted regulation level.	
			1	DCDC output Voltage is above the targeted regulation level.	
31:9	-	RO		Reserved. Read value is undefined.	-

11.5.11 High Speed Crystal Oscillator (16 MHz - 32 MHz) Voltage Source Supply Control register (LDO_XO32 M)

Disclaimer: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

Table 286. High Speed Crystal Oscillator (16 MHz - 32 MHz) Voltage Source Supply Control register (LDO_XO32 M, offset = 0xB0)

Bit	Symbol	Access	Value	Description	Reset value
0	-	W		Reserved Read value is undefined, only zero should be written.	undefined.
1	BYPASS	RW		Activate LDO bypass.	0x0
			0b	Disable bypass mode (for normal operations).	
			1b	Activate LDO bypass.	
2	HIGHZ	RW		Reserved Read value is undefined, only zero should be written.	0x0
			0b	Output in High normal state.	
			1b	Output in High Impedance state.	
5:3	VOUT	RW		Sets the LDO output level.	0x4
			000b	0.750 V.	
			001b	0.775 V.	
			010b	0.800 V.	
			011b	0.825 V.	
			100b	0.850 V.	
			101b	0.875 V.	
			110b	0.900 V.	
			111b	0.925 V.	
7:6	IBIAS	RW		Adjust the biasing current.	0x2
9:8	STABMODE	RW		Stability configuration.	0x3
31:10	-	W	-	Reserved Read value is undefined, only zero should be written.	undefined.

11.5.12 AUX_BIAS

This register controls output of 1V reference voltage.

Table 287. Control output of 1V reference voltage register (AUX_BIAS, offset = 0xB4)

Bit	Symbol	Access	Value	Description	Reset value
0	-	RW		Reserved Read value is undefined, only zero should be written.	-
1	VREF1VENABLE	RW		Control output of 1V reference voltage.	0x0
			0	Output of 1V reference voltage buffer is bypassed.	
			1	Output of 1V reference voltage is enabled.	
31:2	-	RW		Reserved Read value is undefined, only zero should be written.	-

11.5.13 USB High Speed Phy Trim values

Disclaimer: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

This register shows USB High Speed Phy Trim values.

Table 288. USB High Speed Phy Trim values register (USBHS_PHY_TRIM, offset = 0x104)

Bit	Symbol	Access	Value	Description	Reset value
1:0	trim_usb_reg_e nv_tail_adj_vd	RW		Adjusts time constant of HS RX squelch (envelope) comparator.	0x0
5:2	trim_usbphy_tx _d_cal	RW		-	0x0
10:6	trim_usbphy_tx _cal45dp	RW		-	0x0
15:11	trim_usbphy_tx _cal45dm	RW		-	0x0
17:16	trim_usb2_refbi as_tst	RW		-	0x0
20:18	trim_usb2_refbi as_vbgadj	RW		-	0x0
23:21	trim_pll_ctrl0_di v_sel	RW		-	0x0
31:24	-	RW		Reserved Read value is undefined, only zero should be written.	undefined.

11.6 Function description

11.6.1 Frequency measure function

The frequency measure block can be used to measure the frequency of one clock (target clock) using another clock of known frequency (reference clock).

Figure 48 shows a block diagram of the frequency measure function.

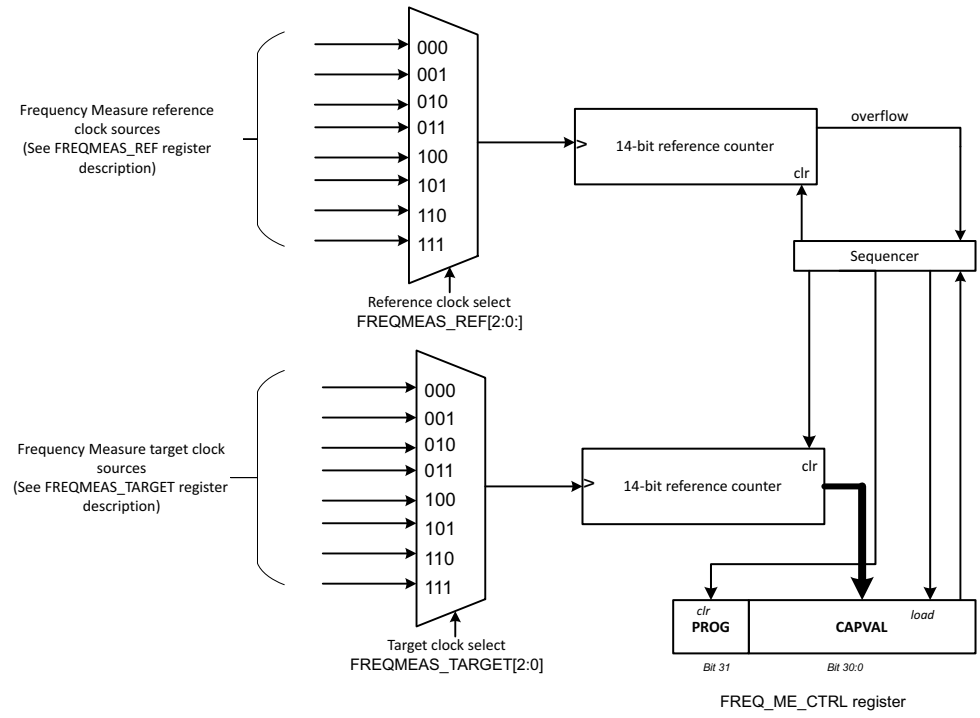


Fig 48. Frequency measure block diagram

Both target and reference clocks are selectable by programming target clock select FREQMEAS_TARGET register and reference clock select FREQMEAS_REF register. See Table 371.

The frequency measure circuit is based on two 31-bit counters, one clocked by the reference clock and one by the target clock. Synchronization between the clocks is performed at the start and end of each count sequence.

A measurement cycle is initiated by software setting the 5 bits SCALE value FREQ_ME_CTRL[4:0], and setting the measurement-in-progress bit FREQ_ME_CTRL[31] in the FREQ_ME_CTRL register. See Table 278.

The software can then poll this same measurement-in-progress PROG bit FREQ_ME_CTRL[31] which will be cleared by hardware when the measurement operation is completed.

The measurement cycle terminates when the reference counter rolls-over, after (2^SCALE-1) reference clock edges. At that point the state of the target counter is loaded into the capture field CAPVAL= FREQ_ME_CTRL[30:0], and the measure-in-progress bit PROG = FREQ_ME_CTRL[31] is cleared.

Software can read this capture value and apply to it a specific calculation which will return the precise frequency of the target clock in MHz. according to following formula:

$F_{target} = (CAPVAL * F_{reference}) / ((1 << SCALE) - 1)$

Example: Reference clock is 1MHz and Target clock is 32K OSC. SCALE = 11

Step 1: FREQMEAS_TARGET[2:0] = 4; FREQMEAS_REF = 3;

Step 2 FREQ_ME_CTRL = (1<<31) + 0xB

Step 3: While (FREQ_ME_CTRL[31] != 0)

Step 4: Read CAPVAL=FREQ_ME_CTRL[30:0]

Step 5: Ftarget = (CAPVAL * 1E6) / ((1<<11)-1)

Remark: Both clocks (reference and target) must be enabled prior to performing this function. If there is a large difference in frequency between the two clocks, then configure the clocks so that the slowest clock is input as Ref Clock. This configuration will provide the highest level of accuracy.

11.6.1.1 Accuracy

The frequency measurement function can measure the frequency of any on-chip (or off-chip) clock (referred to as the target clock) to a high degree of accuracy by using another on-chip clock of known frequency as a reference.

The following constraints apply:

- The frequency of the reference clock must be (somewhat) greater than the frequency of the target clock.
- The system clock used to access the frequency measure function register must also be greater than the frequency of the target clock.

The frequency measurement function circuit is able to measure the target frequency with an error of less than 0.1%, provided the reference frequency is precisely known. Uncertainty in the reference clock (for example the ±1% accuracy of the FRO) will add to the measurement error of the target clock. In general, though, its additional error is less than the uncertainty of the reference clock.
There can also be a modest loss of accuracy if the reference frequency exceeds the target frequency by a very large margin (25x or more). However, accuracy is not a simple function of the magnitude of the frequency difference. Nearly identical frequency combinations, still with a spread of about 43x, result in errors of less than 0.05%. If the target and reference clocks are different by more than a factor of approximately 500,then the accuracy decreases to ±4%.

12.1 How to read this chapter

All LPC55S6 products have two crystal oscillators: A 16 MHz crystal oscillator (also referred as *High Speed* crystal oscillator) and a 32 kHz crystal oscillator (also referred as *Low Speed* crystal oscillator).

Each crystal oscillator has one embedded capacitor bank. The capacitor banks can be used as integrated load capacitors for the crystal oscillators. The capacitor banks on each crystal pins can tune the frequency for crystals with a Capacitive Load (CL) between 6 to 10pF (IEC equivalent). Therefore, they preserve component area on the PCB and Bill of Materials (BOM).

Note: On power-up, a reset value (default value) is applied to each capacitor bank (code 64 dec, xtal_in and amp; xtal_out).

12.2 Features

- Conserves component area on the PCB.
- Save on Bill of Materials (BOM).
- Maintains a Capacitive Load (CL) between 6 - 10 pF (IEC equivalent).
- Relies on simple APIs to configure the Capacitor Banks based on the crystal Capacitive Load (CL) and measured PCB parasitic capacitances on the XIN and XOUT pins.

12.3 Crystal Oscillator Capacitor Banks API description

Table 289. Low power API calls

Function prototype	API description	Section
`void POWER_Xtal16mhzCapbankTrim(int32_t pi32_16MfXtalIecLoadpF_x100, int32_t pi32_16MfXtalPPcbParCappF_x100, int32_t pi32_16MfXtalNPcbParCappF_x100);`	This API configures the Capacitor Bank of the 16 MHz crystal oscillator.	12.3.1
`void POWER_Xtal32mhzCapbankTrim(int32_t pi32_32kfXtalIecLoadpF_x100, int32_t pi32_32kfXtalPPcbParCappF_x100, int32_t pi32_32kfXtalNPcbParCappF_x100);`	This API configures the Capacitor Bank of the 32 kHz crystal oscillator.	12.3.2

12.3.1 XTAL_16mhz_capabank_trim

This API function configures the 16 MHz Crystal Oscillator Capacitor Bank.

Table 290. XTAL_16mhz_capabank_trim API routine

Routine	XTAL_16mhz_capabank_trim
SKD prototype	`void XTAL_16mhz_capabank_trim (int32_t pi32_16MfXtalIecLoadpF_x100, int32_t pi32_16MfXtalPPcbParCappF_x100, int32_t pi32_16MfXtalNPcbParCappF_x100)`
Input parameter	**Param0:** `pi32_16MfXtalIecLoadpF_x100` **Param1:** `pi32_16MfXtalPPcbParCappF_x100` **Param2:** `pi32_16MfXtalNPcbParCappF_x100`
Result	None
Description	Enables the 16 MHz crystal oscillator LDO (voltage regulator) then sets up the Capacitors Banks according to the parameters provided by the user.

Remark: This API does not enable the 16 MHz Crystal Oscillator.

12.3.1.1 Param0: `pi32_16MfXtalIecLoadpF_x100`

The Crystal Oscillator IEC Load capacitance, in pF x 100. For example:

- 6pF IEC equivalent Load Capacitance (which means 12pF on pin XIN and 12pF on pin XOUT) becomes 600.
- 10.2pF IEC Load Capacitance (which means 20.4pF on pin XIN and 20.4pF on pin XOUT) becomes 1020.

12.3.1.2 Param1: `pi32_16MfXtalPPcbParCappF_x100`

PCB parasitic capacitance on pin XIN, in pF x 100. For example:

- 2pF parasitic capacitance becomes 200.
- 0.2pF parasitic capacitance becomes 20.

12.3.1.3 Param2: `pi32_16MfXtalNPcbParCappF_x100`

PCB parasitic capacitance on pin XOUT, in pF x 100. For example:

- 2pF parasitic capacitance becomes 200.
- 0.2pF parasitic capacitance becomes 20.

12.3.2 XTAL_32kHz_capabank_trim

This API function configures the 32 kHz Crystal Oscillator Capacitor Bank.

Table 291. XTAL_32kHz_capabank_trim API routine

Routine	XTAL_32kHz_capabank_trim
SKD prototype	`void XTAL_32kHz_capabank_trim (int32_t pi32_32kfXtalIecLoadpF_x100, int32_t pi32_32kfXtalPPcbParCappF_x100, int32_t pi32_32kfXtalNPcbParCappF_x100)`
Input parameter	**Param0:** `pi32_32kfXtalIecLoadpF_x100` **Param1:** `pi32_32kfXtalPPcbParCappF_x100` **Param2:** `pi32_32kfXtalNPcbParCappF_x100`
Result	None
Description	Sets up the Capacitors Banks according to the parameters provided by the user.

Remark: This API does not enable the 32 kHz Crystal Oscillator.

12.3.2.1 Param0: `pi32_32kfXtalIecLoadpF_x100`

The Crystal Oscillator IEC Load capacitance, in pF x 100. For example:

- 6pF IEC equivalent Load Capacitance (which means 12pF on pin XIN and 12pF on pin XOUT) becomes 600.
- 10.2pF IEC Load Capacitance (which means 5.1pF on pin XIN and 5.1pF on pin XOUT) becomes 1020.

12.3.2.2 Param1: `pi32_32kfXtalPPcbParCappF_x100`

PCB parasitic capacitance on pin XIN, in pF x 100. For example:

- 2pF parasitic capacitance becomes 200.
- 0.2pF parasitic capacitance becomes 20.

12.3.2.3 Param2: `pi32_32kfXtalNPcbParCappF_x100`

PCB parasitic capacitance on pin XOUT, in pF x 100. For example:

- 2pF parasitic capacitance becomes 200.
- 0.2pF parasitic capacitance becomes 20.

12.4 Programming examples

12.4.1 16 MHz Crystal Oscillator

The three variables below are used in all subsequent examples:

int32_t i32_iec_cl_pf; /* IEC equivalent Capacitance Load, in pF */

int32_i i32_xin_pcb_para_pf; /* PCB parasitic capacitance on XIN pin, in pF */

int32_i i32_xout_pcb_para_pf; /* PCB parasitic capacitance on XOUT pin, in pF */

12.4.1.1 Example 1: 8pF IEC Capacitance Load, 2pF PCB parasitic capacitance on XIN pin, 3pF PCB parasitic capacitance on XOUT pin.

i32_iec_cl_pf = 8; /* IEC equivalent Capacitance Load, in pF, which means 16 pF on XIN pin and 16 pF on XOUT pin */

i32_xin_pcb_para_pf = 2; /* PCB parasitic capacitance on XIN pin, in pF */

i32_xout_pcb_para_pf = 3; /* PCB parasitic capacitance on XOUT pin, in pF */

Computation of the required Capacitance Load:

MAXIMUM(2*i32_iec_cl_pf - i32_xin_pcb_para_pf, 2*i32_iec_cl_pf - i32_xout_pcb_para_pf) = MAXIMUM(2*8 - 2, 2*8 - 3) = MAXIMUM(14,13) = 14 pF

14 pF is below 20 pF (10 pF equivalent IEC); therefore, there is no need to add some capacitance on PCB.

Configuration of the internal capa banks:

```
/*
  * - Setup 16-MHz Crystal Oscillator Capacitor Bank and enable 16-MHz Crystal Oscillator LDO (Voltage regulator).
  */
  XTAL_16mhz_capabank_trim(8 * 100, 2 * 100, 3 * 100);
/*
  * - Enable 16-MHz Crystal Oscillator
  */
  PMC->PDRUNCFGCLR0 = PMC_PDRUNCFG0_PDEN_XTAL32M_MASK;
/*
  * - Enable 16-MHz Crystal Oscillator output towards USB High Speed PLL
  */
  ANACTRL->XO32M_CTRL = ANACTRL->XO32M_CTRL | ANACTRL_XO32M_CTRL_ENABLE_PLL_USB_OUT_MASK;
/*
  * - (If required) Enable 16-MHz Crystal Oscillator output for use as System Clock.
  */
  ANACTRL->XO32M_CTRL = ANACTRL->XO32M_CTRL | ANACTRL_XO32M_CTRL_ENABLE_SYSTEM_CLK_OUT_MASK;
```

12.4.1.2 Example 2: 15pF IEC Capacitance Load, 2pF PCB parasitic capacitance on XIN pin, 2pF PCB parasitic capacitance on XOUT pin.

```
i32_iec_cl_pf = 15;  /* IEC equivalent Capacitance Load, in pF, which means 30 pF on XIN pin and 30 pF on XOUT pin */
i32_xin_pcb_para_pf = 2; /* PCB parasitic capacitance on XIN pin, in pF */
i32_xout_pcb_para_pf = 2; /* PCB parasitic capacitance on XOUT pin, in pF */
```

Computation of the required Capacitance Load:

MAXIMUM(2*i32_iec_cl_pf - i32_xin_pcb_para_pf, 2*i32_iec_cl_pf - i32_xout_pcb_para_pf) = MAXIMUM(2*15 - 2, 2*15 - 2) = MAXIMUM(28,28) = 28 pF

28 pF is above 20 pF (10 pF equivalent IEC); therefore, some extra capacitance on PCB are required.

Because some extra capacitance is required on PCB, it is recommended to configure the internal capa bank *as if an **8pF** Load Capacitance IEC equivalent (16pF on both XIN and XOUT pins) was required*, which means:

2*i32_iec_cl_pf - i32_xin_pcb_para_pf must be equal to 16pF.

=> 2*i32_iec_cl_pf = 16 + i32_xin_pcb_para_pf = 16 + 2 = 18 pF (9pF Load Capacitance load IEC equivalent)

=> i32_iec_cl_pf = 18 / 2

=> i32_iec_cl_pf = 9.

Therefore, only 30 pF – 18 pF = 12 pF Load Capacitance is required on the PCB for Xin and XOUT pins.

Configuration of the internal capa banks:

```
 /*
   * - Setup 16 MHz Crystal Oscillator Capacitor Bank and enable 16 MHz Crystal Oscillator LDO (voltage regulator).
   */
   XTAL_16mhz_capabank_trim(9 * 100, 2 * 100, 2 * 100);
 /*
   * - Enable 16 MHz Crystal Oscillator
   */
   PMC->PDRUNCFGCLR0 = PMC_PDRUNCFG0_PDEN_XTAL32M_MASK;
 /*
   * - Enable 16 MHz Crystal Oscillator output towards USB High Speed PLL
   */
   ANACTRL->XO32M_CTRL = ANACTRL->XO32M_CTRL | ANACTRL_XO32M_CTRL_ENABLE_PLL_USB_OUT_MASK;
 /*
   * - (If required) Enable 16 MHz Crystal Oscillator output for use as System Clock.
   */
   ANACTRL->XO32M_CTRL = ANACTRL->XO32M_CTRL | ANACTRL_XO32M_CTRL_ENABLE_SYSTEM_CLK_OUT_MASK;
```

12.4.2 32 kHz Crystal Oscillator

The three variables below are used in all subsequent examples:

```
 int32_t i32_iec_cl_pf;  /* IEC equivalent Capacitance Load, in pF */
int32_i i32_xin_pcb_para_pf; /* PCB parasitic capacitance on XIN pin, in pF */
int32_i i32_xout_pcb_para_pf; /* PCB parasitic capacitance on XOUT pin, in pF */
```

12.4.2.1 Example 1: 8pF IEC Capacitance Load, 2pF PCB parasitic capacitance on XIN pin, 3pF PCB parasitic capacitance on XOUT pin.

i32_iec_cl_pf = 8; /* IEC equivalent Capacitance Load, in pF, which means 16 pF on XIN pin and 16 pF on XOUT pin */

i32_xin_pcb_para_pf = 2; /* PCB parasitic capacitance on XIN pin, in pF */

i32_xout_pcb_para_pf = 3; /* PCB parasitic capacitance on XOUT pin, in pF */

Computation of the required Capacitance Load:

MAXIMUM(2*i32_iec_cl_pf - i32_xin_pcb_para_pf, 2*i32_iec_cl_pf - i32_xout_pcb_para_pf) = MAXIMUM(2*8 - 2, 2*8 - 3) = MAXIMUM(14,13) = 14 pF

14 pF is below 20 pF (10 pF equivalent IEC); therefore, there is no need to add some capacitance on PCB.

Configuration of the internal capa banks:

```
/*
  * - Setup 32 kHz Crystal Oscillator Capacitor Bank
  */
  XTAL_32kHz_capabank_trim(8 * 100, 2 * 100, 3 * 100);
/*
  * - Enable 32 kHz Crystal Oscillator
  */
  PMC->PDRUNCFGCLR0 = PMC_PDRUNCFG0_PDEN_XTAL32K_MASK;
/*
  * - Select 32 kHz Crystal Oscillator (instead of 32 kHz Free Running Oscillator)
  */
  PMC->RTCOSC32K = PMC->RTCOSC32K | PMC_RTCOSC32K_SEL_MASK;
```

12.4.2.2 Example 2: 15pF IEC Capacitance Load, 2pF PCB parasitic capacitance on XIN pin, 2pF PCB parasitic capacitance on XOUT pin.

i32_iec_cl_pf = 15; /* IEC equivalent Capacitance Load, in pF, which means 30 pF on XIN pin and 30 pF on XOUT pin */

i32_xin_pcb_para_pf = 2; /* PCB parasitic capacitance on XIN pin, in pF */

i32_xout_pcb_para_pf = 2; /* PCB parasitic capacitance on XOUT pin, in pF */

Computation of the required Capacitance Load:

MAXIMUM(2*i32_iec_cl_pf - i32_xin_pcb_para_pf, 2*i32_iec_cl_pf - i32_xout_pcb_para_pf) = MAXIMUM(2*15 - 2, 2*15 - 2) = MAXIMUM(28,28) = 28 pF

28 pF is above 20 pF (10 pF equivalent IEC); therefore, some extra capacitance on PCB are required.

Because some extra capacitance is required on PCB, it is recommended to configure the internal capa bank *as if an 8pF Load Capacitance IEC equivalent (16pF on both XIN and XOUT pins) was required*, which means:

2*i32_iec_cl_pf - i32_xin_pcb_para_pf must be equal to 16pF.

=> 2*i32_iec_cl_pf = 16 + i32_xin_pcb_para_pf = 16 + 2 = 18 pF (9pF Load Capacitance load IEC equivalent)

=> i32_iec_cl_pf = 18 / 2

=> i32_iec_cl_pf = 9.

Therefore, only 30 pF – 18 pF = 12 pF Load Capacitance is required on the PCB for Xin and XOUT pins.

Configuration of the internal capa banks:

```
/*
  * - Setup 32 kHz Crystal Oscillator Capacitor Bank
  */
  XTAL_32kHz_capabank_trim(9 * 100, 2 * 100, 2 * 100);
 /*
  * - Enable 32 kHz Crystal Oscillator
  */
  PMC->PDRUNCFGCLR0 = PMC_PDRUNCFG0_PDEN_XTAL32K_MASK;
 /*
  * - Select 32 kHz Crystal Oscillator (instead of 32 kHz Free Running Oscillator)
  */
  PMC->RTCOSC32K = PMC->RTCOSC32K | PMC_RTCOSC32K_SEL_MASK;
```

13.1 How to read this chapter

This chapter provides an overview of power related information for LPC55S6x/LPC55S2x/LPC552x devices. These devices include a variety of power and clock switches to fine tune power usage to match the requirements for different performance levels and reduced power modes.

To turn analog components ON or OFF in active mode, use the PMC_PDRUNCFG0 register, see Table 311. In deep-sleep, power-down and deep-power down modes, the power profile API controls which analog peripherals remain powered up. See Section 13.3 “Functional description”.

13.2 General description

Usage of CPU0 and CPU1 - low power modes:

- CPU0: can be put in sleep mode. Only CPU0 can trigger deep-sleep, power-down, and deep power-down modes.
- CPU1: can be put in sleep mode. CPU1 can not trigger deep-sleep, power-down, and deep power-down modes.

Also, CPU0 can be put in sleep mode while CPU1 is active.

13.2.1 Power supplies

Power to the part is supplied via seven on-chip regulators:

- Bulk DCDC Converter
- LDO_AO regulator (Always On power domain regulator).
- LDO_MEM regulator (SRAM regulator).
- LDO_USB_HS regulator (USB high speed PHY regulator).
- LDO_FLASH_NV regulator (flash regulator)
- LDO_XO_32M regulator.

All six previously mentioned internal regulators are supplied by the main external supply called VBAT (1.8 V – 3.6 V).

13.2.2 Power domains

The device is partitioned into five power domains:

- PD_CORE: Power Domain *Core*: most of all digital core logic (CPU0, CPU1, multilayer matrix, and almost all peripherals like Flexcomm, SDMA, Power Quad, etc.,).
- PD_SYSTEM: Power Domain *System:* Some critical system components like clocks controller, reset controller and Syscon.

- PD_AO: Power Domain *Always On*: Power management controller, RTC, and OS Event Timer.This domain always has power as long as sufficient voltage is available on VBAT ([1.8 V – 3.6 V]).
- PD_MEM_0: First Power Domain *Memories*: Two 4 KB SRAM instances.
- PD_MEM_1: Second Power Domain *Memories*: All other SRAM instances.

Table 292 shows the detailed list of all modules per power domain.

Table 292. Power domain supply

	Power domain core	Power domain system	Power domain AO	Power domain _mem_0	Power domain _mem_1
Input / Output	-	GPIOs other than the four wake-up GPIO pins.	Four wake-up GPIO pins.	-	-
Analog components	PLL0	[1]	32-kHz Crystal Oscillator (XTAL32K)		
	PLL1		32-kHz Free Running Oscillator (FRO32K)		
	192-MHz Free Running Oscillator (FRO192M)		1-MHz Free Running Oscillator (FRO1M)		
	32-MHz Crystal Oscillator		Analog Comparator		
	ADC		Brown Out Detector VBAT (BoD VBAT)		
	USB High Speed Physical Interface				
	Temperature Sensor				
	Brown Out Detector Core (BoD Core)				

Table 292. Power domain supply *…continued*

	Power domain core	Power domain system	Power domain AO	Power domain _mem_0	Power domain _mem_1
Digital components	CPU0 (Cortex-M33 full feature)	Reset controller	Power Management Controller (PMC)		
	CPU1 (Cortex-M33 light)	System Configuration (Syscon)	Real Time Clock (RTC)		
	Debug Ports (SWD interface)	I/O Configuration (IOCON)	OS Event Timer		
	Mailbox	Sleep controller			
	Debug Mailbox				
	-	Group GPIO Input Interrupt (GINT0/1)			
	Power Quad	Secret keys (extracted from PUF)			
	CASPER	Flexcomm Interface3			
	PRINCE				
	SDIO				
	System DMA				
	Secure System DMA				
	General Purpose I/O Controller				
Digital components	Secure General Purpose I/O Controller				
	High Speed SPI				

Table 292. Power domain supply *…continued*

	Power domain core	Power domain system	Power domain AO	Power domain _mem_0	Power domain _mem_1
	Flexcomm 0,1,2,4,5,6,7				
	USB Full Speed				
	USB High Speed				
	Hash/AES Crypto Engine				
	CRC Engine				
	SCTimer/PWM (SCT)				
	Standard Counter/Timer 0,1,2,3,4 (CTIMER)				
	ROM/SRAM/Flash Controllers				
	Micro-Tick Timer				
	Mutli Rate Timer (MRT)				
	Widowed-Watchdog Timer (WWDT)				
	True Random Number Generator				
	Programmable Logic (PLU/LUT)				
	Physically Unclonable Function (PUF)				
	Analog Controller				
	Peripheral Input Multiplexing				
	Pin Interrupt & Pattern Matching (PINT)				
	ADC Controller				
Memories	ROM (128 KB)			RAM_X2 (4 KB)	RAM_X0 (16 KB)
	FLASH (640 KB)			RAM_X3 (4 KB)	RAM_X1 (8 KB)
	PUF RAM (2 KB)				RAM_00 (32 KB)
					RAM_01 (32 KB)
					RAM_10 (64 KB)
					RAM_20 (64 KB)

Table 292. Power domain supply *…continued*

	Power domain core	Power domain system	Power domain AO	Power domain _mem_0	Power domain _mem_1
					RAM_30 (32 KB)
					RAM_31 (32 KB)
					RAM_40 (4 KB)
					RAM_41 (4 KB)
					RAM_42 (4 KB)
					RAM_43 (4 KB)
					RAM_USB (16 KB)

[1] The GPIO logic level does not remain static in power-down mode. All GPIO pin state will be logic '0' in power-down mode.

13.2.3 Power modes

Power modes are controlled exclusively via a power API Chapter 14 "LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API" and the operating mode of the CPU.

There are five power modes, from highest to lowest power consumption:

- Active: The part is in *Active mode* after a Power-On Reset (POR), hardware pin reset or software reset and when it is fully powered.
- Sleep: *Sleep-mode* saves a significant amount of power by stopping CPU execution without affecting peripherals or requiring significant wake-up time. The sleep-mode affects the relevant CPU only. The clock to the CPU is shut off. Peripherals and memories are active and operational.
- Deep-sleep mode: Deep-sleep mode is configurable. The full IC remained powered, but flash and ROM are shut down, with the cost of a longer wake-up time compared to the sleep-mode. The clock to all CPUs is shut down; if not configured, the peripherals receive no internal clocks. All SRAM, GPIO logic, and registers maintain their internal states. All SRAM instances that are not configured to enter in *retention state* will stay in active mode and therefore consume more power. Some peripherals can have DMA service during deep-sleep mode without waking up entire device. Through the power profiles API, selected peripherals such as USB, SPI, I^2C, USART, WWDT, RTC, Counter/Timers, and BOD can be left running in deep-sleep mode.
- Power-down mode: Power-down mode turns off nearly all on-chip power consumption by shutting down the DCDC, with the cost of a longer wake-up time compared to deep-sleep mode. The power-down mode affects the entire system, the clock to all CPUs and peripheral is shut down and, if not configured, the peripherals in power domains PD_SYSTEM and PD_AO receive no internal clocks. All SRAM can be configured to maintain their internal state and all registers lose their internal states except those located in the power domains PD_SYSTEM and PD_AO. Any SRAM instance that is not configured to maintain its internal state will lose it. The internal state of the CPU0, ROM patch unit, AHB security controller and PRINCE are maintained. When a wake-up event occurs, code execution will resume from where it left off. It is the responsibility of the customer application to re-configure all modules in the power domain core PD_CORE (whose states have not been retained (i.e., SDMA,

PowerQuad, and all Flexcomm products except for Flexcomm3). Through the power profiles API, selected peripherals such as FLEXCOMM3 Interface, SPI, I^2C, USART, GINTO, RTC, OS event timer or analog comparator, can be enabled to wake-up the system.

- Deep power-down: Deep-power down mode shuts down virtually all on-chip power consumption but requires a significantly longer wake-up time (compared to power-down mode). For maximal power savings, the entire system (CPUs and all peripherals) is shut down except for the PMU, the PMC, the RTC and the OS event timer. On wake-up, the part reboots.

The table below summarizes the power state of the different power domains according to the power modes.

Table 293. Power modes

	PD_CORE	PD_SYSTEM	PD_AO	PD_MEM_0	PD_MEM_1
ACTIVE	ON	ON	ON	ON	ON
SLEEP	ON	ON	ON	ON	ON
DEEP-SLEEP	ON	ON	ON	ON/OFF	ON/OFF
POWER-DOWN	OFF	ON	ON	ON/OFF	ON/OFF
DEEP POWER- DOWN	OFF	OFF	ON	ON/OFF	ON/OFF

13.2.4 Peripheral configuration in reduced power modes

Table 294. Peripheral reduced power modes

Peripherals		Reduced Power Modes			
Name	**Description**	**Sleep**	**Deep-sleep**	**Power-down**	**Deep-power down**
DCDC	Bulk DCDC Converter	ON	ON	OFF	OFF
RTC	Real-time Clock	Software configured	Software configured	Software configured	Software configured
BIAS	Analog references	ON	Software configured	Software configured	OFF
BoD VBAT	VBAT Brown Out Detector	Software configured	Software configured	OFF	OFF
FRO1M	1 MHz Free Running Oscillator	ON	Software configured	OFF	OFF
FRO192M	192 MHz Free Running Oscillator. This provides the 12 MHz FRO (divided down from the currently selected on-chip FRO_192 oscillator).	ON	Software configured	OFF	OFF
FRO32K	32 kHz Free Running Oscillator	Software configured	Software configured	Software configured	Software configured
XTAL32K	32 kHz Crystal Oscillator	Software configured	Software configured	Software configured	Software configured
XTAL32M	32 MHz Crystal Oscillator	Software configured	Software configured	OFF	OFF

Table 294. Peripheral reduced power modes *…continued*

Peripherals		Reduced Power Modes			
Name	**Description**	**Sleep**	**Deep-sleep**	**Power-down**	**Deep-power down**
PLL0	1st PLL	Software configured	Software configured	OFF	OFF
PLL1	2nd PLL	Software configured	Software configured	OFF	OFF
USB_FS_PHY	USB Full Speed Physical	Software configured	Software configured	OFF	OFF
USB_HS_PHY	USB High Speed Physical	Software configured	Software configured	OFF	OFF
COMP	Analog comparator	Software configured	Software configured	Software configured	OFF
TEMPSENS	Temperature Sensor	Software configured	Software configured	OFF	OFF
ADC	General Purpose ADC	Software configured	Software configured	OFF	OFF
LDO_MEM	SRAM Regulator	OFF	ON	Software configured	Software configured
LDO_DEEP_SLEEP	SRAM Regulator	Software configured	Software configured	OFF	OFF
LDO_USB_HS	USB High Speed Regulator	Software configured	Software configured	OFF	OFF
AUXBIAS	ADC Analog references (BIAS_VREF_IV)	Software configured	Software configured	OFF	OFF
LDO_XO_32M	32 MHz Crystal Oscillator Regulator	Software configured	Software configured	OFF	OFF
LDO_FLASH_NV	Flash Regulator	ON	OFF	OFF	OFF
RNG	True Random Number Generator	Software configured	Software configured	OFF	OFF
PLL0_SSCG	PLL0 Spread Spectrum Clock Generator	Software configured	Software configured	OFF	OFF
ROM	ROM	ON	OFF	OFF	OFF

13.2.5 Wake-up process

The part always wakes up to the active mode. To wake up from the reduced power modes, you must configure the wake-up source. Each reduced power mode supports its own wake-up sources and needs to be configured accordingly. See Table 323 “Parameter wakeup_interrupts”.

13.3 Functional description

13.3.1 Power management

The LPC55xx support a variety of power control features. In active mode, when the chip is running, power and clocks to selected peripherals can be optimized for power consumption. In addition, there are four special modes of processor power reduction with different peripherals running: sleep-mode, deep-sleep mode, power-down and deep-power down mode, activated by the power-mode configure API, see Chapter 14 “LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API”.

Remark: The debug mode is not supported in sleep, deep-sleep, power-down or deep-power down modes.

13.3.2 Active mode

In Active mode, the CPU, memories, and peripherals are clocked by the AHB/CPU clock.

The chip is in active mode after reset and the default power configuration is determined by the reset values of the PDRUNCFG0, AHBCLKCTRL0, and AHBCLKCTRL1 registers. The power configuration can be changed during run time.

13.3.2.1 Power configuration in active mode

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

- The AHBCLKCTRL registers control which memories and peripherals are enabled. See Section 4.5.17 “AHB clock control 0”, Section 4.5.18 “AHB clock control 1”, and Section 4.5.19 “AHB clock control 2”. To save power, functions that are not needed by the application should be turned off. If specific times are known when certain functions will not be needed, they can be turned OFF temporarily and turned back ON again as needed.
- The power to various analog blocks (PLL, oscillators, and the BOD circuit) can be controlled individually through the PDRUNCFG0 register, see Table 311. As with clock controls, these blocks should generally be turned OFF if not needed by the application. If turned OFF, time will be needed before these blocks can be used again after being turned ON.
- The clock source for the system clock can be selected from the FRO (default), the 32 kHz oscillator, the 1-MHz FRO, the 16 MHz crystal oscillator or an external clock, see Figure 4 and related registers.
- The system clock frequency can be selected, see Section 4.2.3 “Configure the main clock and system clock” and other clocking related sections. Generally speaking, everything uses less power at lower frequencies, so running the CPU and other device features at a frequency sufficient for the application (plus some margin) will save power. If the PLL is not needed, it should be turned OFF to save power. Also, running the PLL at a lower CCO frequency saves power.
- Several peripherals use individual peripheral clocks with their own clock dividers. The peripheral clocks can be shut down through the corresponding clock divider registers if the base clock is still needed for another function.

- The power library provides an easy way to optimize power consumption depending on CPU load and performance requirements. See Chapter 14 “LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API”.

13.3.3 Sleep-mode

In sleep-mode, the system clock to the CPU 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 AHBCLKCTRL registers, 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 are maintained.

As in active mode, the power API provides an easy way to optimize power consumption depending on CPU load and performance requirements in sleep-mode. See Chapter 14 “LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API”.

13.3.3.1 Power configuration in power mode

Power consumption in sleep-mode is configured by the same settings as in active mode:

- Enabled clocks remain running.
- The system clock frequency remains the same as in active mode, but the processor is not clocked.
- Analog and digital peripherals are powered and selected as in active mode through the PDRUNCFG0, AHBCLKCTRL0, AHBCLKCTRL1, and AHBCLKCTRL2 registers.

13.3.3.2 Programming sleep-mode

The following steps must be performed to enter sleep-mode

1. In the NVIC, enable all interrupts that are needed to wake-up the relevant CPU.
2. Alter PMC->PDRUNCFG0 if needed to reflect any functions that should be ON or OFF during sleep-mode.
3. Call the power API CHIPLOWPOWER_enter_sleep(). See Chapter 14 “LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API”.

13.3.3.3 Wake-up from sleep-mode

A CPU sleep-mode is exited automatically when an interrupt enabled by the NVIC arrives at the processor or a reset occurs. After wake-up caused by an interrupt, the device returns to its original power configuration defined by the contents of the PDRUNCFG0 and the AHBCLKCTRL registers. If a reset occurs, the micro-controller enters the default configuration in active mode.

13.3.4 Deep-sleep mode

In deep-sleep mode, the system clock to the CPUs are disabled as in sleep-mode. Analog blocks are powered down by default but can be selected to keep running through the power API if needed as wake-up sources. The main clock and all peripheral clocks are

disabled. The FRO 1 MHz and the FRO 192 MHz can be disabled. The flash memory and ROM are put in shutdown mode.

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

GPIO pin interrupts, GPIO group interrupts, and selected peripherals such as USB Full Speed and USB High Speed, SPI, I^2C, USART, WWDT, RTC, standard Counter/Timers, and BOD can be left running in deep-sleep mode. The FRO1 MHz and the FRO 192 MHz, RTC oscillator, and the watchdog oscillators (FRO 32 kHz and Crystal 32-kHz).

In some cases, DMA can operate in deep-sleep mode.

13.3.4.1 Power configuration in deep-sleep mode

Power consumption in deep-sleep mode is determined primarily by which analog wake-up sources remain enabled. Serial peripherals and pin interrupts configured to wake-up the part, contribute to the power consumption only to the extent that they are clocked by external sources. All wake-up events (other than reset) must be enabled through the power API. In addition, any related analog block, for example: the RTC oscillators or low power 1-MHz FRO must be explicitly enabled through a power API function. See Chapter 14 “LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API”.

13.3.4.2 Programming deep-sleep mode

The following step must be performed to enter deep-sleep mode:

1. On power-up, the BOD VBAT, and BOD CORE are enabled. Power API disables BOD CORE reset and interrupt generation in deep-sleep mode.
2. Call the power API with the peripheral parameter to enable the analog peripherals and wake-up sources/events, See Chapter 14 “LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API”.

13.3.4.3 Wake-up from deep-sleep mode

The part can wake-up from deep-sleep mode in the following ways:

- Using a signal on one of the eight pin interrupts selected. See Chapter 19 “LPC55S6x/LPC55S2x/LPC552x Pin Interrupt and Pattern Match (PINT)”. Each pin interrupt must also be enabled via the power API.
- Using an interrupt from a block such as the watchdog interrupt or RTC interrupt, when enabled during the reduced power mode via the power API. Also enable the wake-up sources in power API.
- Using a reset from the RESET pin, or WWDT (if enabled via the power API).
- Using a wake-up signal from any of the serial peripherals that are operating in deep-sleep mode. Also enable the wake-up sources via the power API.
- GPIO group interrupt signal. The interrupt must also be enabled via the power API.
- RTC alarm signal or wake-up signal. See Chapter 28 “LPC55S6x/LPC55S2x/LPC552x Real-Time Clock (RTC)”. Interrupts must also be enabled via the power API.

- OS Event Timer. See Chapter 31 “LPC55S6x/LPC55S2x/LPC552x OS Event Timer”. Interrupts must also be enabled via the power API.

13.3.5 Power-down mode

In power-down mode, nearly all on-chip power consumption is turned off by shutting down the internal DC-DC converter. FRO 192 MHz and FRO 1 MHz are disabled. The flash is powered down. The system clock to the CPU is stopped and if not configured, the peripherals receives no clocks. Through the power profiles API, selected peripherals such as Flexcomm interfaces 3 (SPI, I2C, USART, I2S), RTC, OS Timer, and comparator can be left running in power-down mode. Clock sources such as FRO 32 kHz, and the 32.768 kHz RTC clock can be enabled or disabled via software.

The chip can wake up from power-down mode via a reset, digital pins selected as inputs to the group interrupt block, OS Timer, RTC alarm, an interrupt from the Flexcomm Interface 3 (SPI, I2C, I2S, USART), and comparator. In power-down mode, the CPU processor state is retained to allow resumption of code execution when a wake-up event occurs.

All SRAM can be configured to maintain their internal state as long as it is configured to do so using power API call. The GPIO logic level does not remain static in power-down mode. All GPIO pin state will be logic '0' in power-down mode.

All IOCON registers and peripheral registers related ONLY to Flexcomm3 (SPI, I2C, I2S, USART), GINTO0, RTC, OS Event timer, and analog comparator will maintain state.

13.3.5.1 Power configuration in power-down mode

Power consumption in power-down mode is determined primarily by the number of SRAM instances which remain enabled (retention mode). Serial peripherals in Flexcomm3 and pin interrupts configured to wake-up contribute to the dynamic power consumption only to the extent that they are clocked by external sources. All wake-up events (other than reset) must be enabled via the power API. In addition, any analog block (the analog comparator, the 32-kHz XTAL and 32-kHz FRO) must be explicitly enabled through a power API function, See Chapter 14 “LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API”.

13.3.5.2 Programming power-down mode

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

1. Enable the CPU retention mode via the Power API.
2. On power-up, the BODs are enabled. Power API disables BODs in power-down mode and restores the configuration after wake-up from power-down.
3. Call the power API with the peripheral parameter to enable the analog peripherals (analog comparator, 32-kHz XTAL or 32-kHz FRO) and select the wake-up sources. See Chapter 14 “LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API”.

13.3.5.3 Wake-up from power-down mode

The part can wake-up from power-down mode in the following ways:

- Using a reset from the RESET pin.
- Using a wake-up signal from any of the serial peripherals in Flexcomm3. Also enable the wake-up sources via the power API.
- Using the analog comparator. Also enable the wake-up sources via the power API.
- GPIO group interrupt signal. The interrupt must also be enabled via the power API.
- RTC alarm signal or wake-up signal. See Chapter 28 "LPC55S6x/LPC55S2x/LPC552x Real-Time Clock (RTC)". Interrupts must also be enabled via the power API.
- OS Event Timer. See Chapter 31 "LPC55S6x/LPC55S2x/LPC552x OS Event Timer". Interrupts must also be enabled via the power API.

13.3.6 Deep power-down mode

In deep-power down mode, power and clocks are shut off to the entire chip with the exception of the PMC, the RTC and the OS Event Timer.

During deep-power down mode, the contents of the SRAM can be retained (software configured via the low power API) and registers (other than those in the PMC, the RTC and OS Event Timer) are not retained. All functional pins are tri-stated in deep-power down mode, except the four wake-up pins and the RESET pin.

13.3.6.1 Power configuration in deep power-down mode

Deep power-down mode has the following configuration options (via the low power API)

- RAMs instances to be retained.
- Wake-up pins.
- 32 kHz clock source for RTC and OS Event Timer.

All clocks, the core, and all peripherals are powered down. Only the PMC, RTC and OS Event Timer are powered with the associated clock source: 32 kHz FRO or 32 kHz crystal.

13.3.6.2 Wake-up from deep power-down mode

Wake-up from deep-power down can be accomplished via:

- The RESET pin.
- The RTC.
- The OS Event Timer.
- The four wake-up pins.

13.3.6.3 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. It starts the high-resolution timer if enabled. Another option is to use the 1Hz alarm timer.
2. Call the power API function, see Chapter 14 "LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API".

13.3.6.4 Programming deep-power down mode using the OS Event Timer for wake-up

The following steps must be performed to enter deep-power down mode when using the OS Event Timer for waking up:

1. Configure the OS Event Timer clock sources in PMC->OSTIMER.
2. Configure OS Event Timer (OSTIMER->MATCHN, OSTIMER->EVENT_CTRL …).
3. Call the power API function, see Chapter 14 “LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API”.

13.3.6.5 Programming deep-power down mode using the wake-up pins for wake-up

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

1. Call the power API function, see Chapter 14 “LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API”.

13.3.6.6 Wake-up from deep-power down mode

The part goes through the entire reset process when a deep-power down wake-up event occurs:

- 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 will be in their reset state.

13.4 Register description

Table 295. Register overview: pmc (base address = 0x40020000)

Name	Access	Offset	Description	Reset value	Section
STATUS	WO	0x4	Power Management Controller FSM (Finite State Machines) status.	0x0	13.4.1
RESETCTRL	RW	0x8	Reset control (Reset by: PoR, Pin Reset, Brown Out Detectors Reset, deep power- down reset, software reset).	0x0	13.4.2
DCDC0	RW	0x10	DCDC (first) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset].	0x010C 2C68	13.4.3
DCDC1	RW	0x14	DCDC (second) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset].	0x0180 3A98	13.4.4
LDOPMU	RW	0x1C	Power Management Unit (PMU) and Always-On domains LDO control [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset].	0x010E F718	13.4.5
BODVBAT	RW	0x30	VBAT Brown Out Detector (BoD) control register (Reset by: PoR, pin reset, software reset).	For device revision 0A: 0x69. For device revision 1B: 0x47	13.4.6
REFFASTWKUP	RW	0x40	Analog References fast wake-up Control register [Reset by: PoR].	0x1	13.4.7
XTAL32K	RW	0x4C	32 KHz Crystal oscillator (XTAL) control register [Reset by: PoR, Brown Out Detectors Reset].	0xB	13.4.8
COMP	RW	0x50	Analog comparator control register (Reset by: PoR, pin reset, Brown Out Detectors reset, deep power- down reset, software reset).	0xA	13.4.9
WAKEUPIOCTRL	RW	0x64	Deep Power Down wake-up source [Reset by: PoR, Pin Reset, Software Reset].	0x0	13.4.10
WAKEIOCAUSE	RW	0x68	Allows to identify the Wake-up I/O source from deep-power down mode.	0x0	13.4.11
STATUSCLK	RW	0x74	FRO and XTAL status register (Reset by: PoR, Brown Out Detectors reset).	0x6	13.4.12
AOREG1	RW	0x84	General purpose always on domain data storage. **Remark:** This register is managed and updated by the ROM boot and cannot be updated by any application.	0x0	13.4.13
MISCCTRL	RW	0x90	Dummy Control bus to PMU [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset].		13.4.14
RTCOSC32K	RW	0x98	RTC clock control register (Reset by: PoR, Brown Out Detectors reset).	0x03FF0008	13.4.15
OSTIMER	RW	0x9C	OS timer control register (Reset by: PoR, Brown Out Detectors reset).	0x8	13.4.16

Table 295. Register overview: pmc (base address = 0x40020000) *…continued*

Name	Access	Offset	Description	Reset value	Section
PDRUNCFG0	RW	0xB8	Controls the power to various analog blocks (Reset by: PoR, pin reset, Brown Out Detectors reset, deep power-down reset, software reset).	0xDEFFC4	13.4.17
PDRUNCFGSET0	W	0xC0	Controls the power to various analog blocks (Reset by: PoR, pin reset, Brown Out Detectors reset, deep power-down reset, software reset).	0x0	13.4.18
PDRUNCFGCLR0	W	0xC8	Controls the power to various analog blocks (Reset by: PoR, pin reset, Brown Out Detectors reset, deep power-down reset, software reset).	0x0	13.4.19
SRAMCTRL	RW	0xD4	All SRAMs common control signals [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Software Reset].	0x1	13.4.20

13.4.1 Power Management Controller FSM (Finite State Machines) status (STATUS)

Disclaimer: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

This register shows Power Management Controller FSM status.

Table 296. Power Management Controller FSM status (STATUS, offset = 0x4)

Bit	Symbol	Value	Description	Reset value
2-0	FSMMAIN		Power Management Controller Main Finite State Machine (FSM) status.	0x0
		000b	POWER UP: The IC is powering up.	
		001b	ACTIVE: Power up is completed. The IC is in normal functional operation mode.	
		010b	POWER DOWN: the IC has entered POWER DOWN mode.	
		011b	DEEP SLEEP: The IC has entered DEEP SLEEP mode.	
		110b	DEEP POWER DOWN: The IC entered DEEP POWER DOWN mode.	
		111b	IC Structural TEST Mode: The IC has entered in IC Test mode.	
6-3	FSMPWUP		POWER UP Finite State Machine (FSM) status.	0x0
10-7	FSMDSLP		DEEP SLEEP Finite State Machine (FSM) status.	0x0
14-11	FSMPWDN		POWER DOWN Finite State Machine (FSM) status.	0x0
17-15	FSMDPWD		DEEP POWER DOWN Finite State Machine (FSM) status.	0x0
19-18	BOOTMODE		Latest IC Boot cause:	0x0
		00b	Latest IC boot was a Full power cycle boot sequence (PoR, Pin Reset, Brown Out Detectors Reset, Software Reset).	
		01b	Latest IC boot was from DEEP SLEEP low power mode.	
		10b	Latest IC boot was from POWER DOWN low power mode.	
		11b	Latest IC boot was from DEEP POWER DOWN low power mode.	
27-20	-		Reserved Read value is undefined, only zero should be written.	undefined.
31-28	WAFERTESTDONEVECT		Indicates current status of wafer test level.	0x0

13.4.2 Reset control register

Table 297. Reset control (Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset) (RESETCTRL, offset = 0x8)

Bit	Symbol	Value	Description	Reset value
0	DPDWAKEUPRESETENABLE		Wake-up from deep-power down reset event (either from wake up I/O or RTC or OS event timer).	0x0
		0b	Reset event from deep-power down mode is disable.	
		1b	Reset event from deep-power down mode is enable.	
1	BODVBATRESETENABLE		BOD VBAT reset enable.	0x0
		0b	BOD VBAT reset is disable.	
		1b	BOD VBAT reset is enable.	
2	BODCORERESETENABLE		BOD CORE reset enable.[1]	0x0
		0b	BOD CORE reset is disable.	
		1b	BOD CORE reset is enable.	
3	SWRRESETENABLE		Software reset enable.	0x0
		0b	Software reset is disable.	
		1b	Software reset is enable.	
31:4	-		Reserved. Read value is undefined, only zero should be written.	undefined

[1] Disclaimer: The value written to this bit should not be changed. This bit is handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

13.4.3 DCDC (first) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (DCDC0)

Disclaimer: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

This shows DCDC (first) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset].

Table 298. DCDC (first) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (DCDC0, offset = 0x10)

Bit	Symbol	Value	Description	Reset value
5-0	RC		Constant On-Time calibration.	0x28
7-6	ICOMP		Select the type of ZCD comparator.	0x1
9-8	ISEL		Alter Internal biasing currents.	0x2
10	ICENABLE		Selection of auto scaling of COT period with variations in VDD.	0x1
15-11	TMOS		One-shot generator reference current trimming signal.	0x9
16	DISABLEISENS E		Disable Current sensing.	0x0
20-17	VOUT		Set output regulation voltage.	0x6
		0000b	0.95 V.	
		0001b	0.975 V.	
		0010b	1 V.	
		0011b	1.025 V.	
		0100b	1.05 V.	
		0101b	1.075 V.	
		0110b	1.1 V.	
		0111b	1.125 V.	
		1000b	1.15 V.	
		1001b	1.175 V.	
		1010b	1.2 V.	
21	SLICINGENAB LE		Enable staggered switching of power switches.	0x0
22	INDUCTORCLA MPENABLE		Enable shorting of Inductor during PFM idle time.	0x0
26-23	VOUT_PWD		Set output regulation voltage during Deep Sleep.	0x2
31-27	-		Reserved Read value is undefined, only zero should be written.	undefined.

13.4.4 DCDC (second) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (DCDC1)

Disclaimer: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

This shows DCDC (second) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset].

Table 299. DCDC (second) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (DCDC1, offset = 0x14)

Bit	Symbol	Value	Description	Reset value
3-0	RTRIMOFFET		Adjust the offset voltage of BJT based comparator.	0x8
7-4	RSENSETRIM		Adjust Max inductor peak current limiting.	0x9
8	DTESTENABLE		Enable Digital test signals.	0x0
10-9	SETCURVE		Bandgap calibration parameter.	0x1
14-11	SETDC		Bandgap calibration parameter.	0x7
17-15	DTESTSEL		Select the output signal for test.	0x0
18	SCALEENABLE		Modify COT behavior.	0x0
19	FORCEBYPASS		Force bypass mode.	0x0
23-20	TRIMAUTOCOT		Change the scaling ratio of the feedforward compensation.	0x8
24	FORCEFULLC YCLE		Force full PFM PMOS and NMOS cycle.	0x1
25	LCENABLE		Change the range of the peak detector of current inside the inductor.	0x0
30-26	TOFF		Constant Off-Time calibration input.	0x0
31	TOFFENABLE		Enable Constant Off-Time feature.	0x0

13.4.5 Power Management Unit (PMU) and Always-On domains LDO control [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (LDOPMU)

Disclaimer: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

This shows Power Management Unit (PMU) and Always-On domains LDO control [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (LDOPMU) values.

Table 300. Power Management Unit (PMU) and Always-On domains LDO control [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (LDOPMU) (LDOPMU, offset = 0x1C)

Bit	Symbol	Value	Description	Reset value
4-0	VADJ		Sets the Always-On domain LDO output level.	0x18
		00000b	1.22 V.	
		00001b	0.7 V.	
		00010b	0.725 V.	
		00011b	0.75 V.	
		00100b	0.775 V.	
		00101b	0.8 V.	
		00110b	0.825 V.	
		00111b	0.85 V.	
		01000b	0.875 V.	
		01001b	0.9 V.	
		01010b	0.96 V.	
		01011b	0.97 V.	
		01100b	0.98 V.	
		01101b	0.99 V.	
		01110b	1 V.	
		01111b	1.01 V.	
		10000b	1.02 V.	
		10001b	1.03 V.	
		10010b	1.04 V.	
		10011b	1.05 V.	
		10100b	1.06 V.	
		10101b	1.07 V.	
		10110b	1.08 V.	
		10111b	1.09 V.	
		11000b	1.1 V.	
		11001b	1.11 V.	
		11010b	1.12 V.	

Table 300. Power Management Unit (PMU) and Always-On domains LDO control [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (LDOPMU) (LDOPMU, offset = 0x1C)

Bit	Symbol	Value	Description	Reset value
		11011b	1.13 V.	
		11100b	1.14 V.	
		11101b	1.15 V.	
		11110b	1.16 V.	
		11111b	1.22 V.	
9-5	VADJ_PWD		Sets the Always-On domain LDO output level in all power down modes.	0x18
14-10	VADJ_BOOST		Sets the Always-On domain LDO Boost output level.	0x1D
19-15	VADJ_BOOST_ PWD		Sets the Always-On domain LDO Boost output level in all power down modes.	0x1D
23-20	-		Reserved Read value is undefined, only zero should be written.	0x0
24	BOOST_ENA		Control the LDO AO boost mode in ACTIVE mode.	0x1
		0b	LDO AO Boost Mode is disable.	
		1b	LDO AO Boost Mode is enable.	
25	BOOST_ENA_ PWD		Control the LDO AO boost mode in the different low power modes (DEEP SLEEP, POWERDOWN, and DEEP POWER DOWN).	0x0
		0b	LDO AO Boost Mode is disable.	
		1b	LDO AO Boost Mode is enable.	
31-26	-		Reserved Read value is undefined, only zero should be written.	undefined.

13.4.6 VBAT Brown Out Detector (BoD) control register

Brown-Out Detector (BOD) for VBAT_DCDC voltage with separate thresholds for interrupt and forced reset can be programmed using the VBAT BOD control register.

Table 301. VBAT Brown Out Detector (BoD) control register (Reset by: PoR, pin reset, software reset) (BODVBAT, offset = 0x30)

Bit	Symbol	Value	Description	Reset value
4:0	TRIGLVL		BoD trigger level.	For device revision 0A: 0x9. For device revision 1B, hardware default value: 0x7. After ROM execution, value is set to 0x9.
		0	1.00 V.	
		1	1.10 V.	
		2	1.20 V.	
		3	1.30 V.	
		4	1.40 V.	
		5	1.50 V.	
		6	1.60 V.	
		7	1.65 V.	
		8	1.70 V.	
		9	1.75 V.	
		10	1.80 V.	
		11	1.90 V.	
		12	2.00 V.	
		13	2.10 V.	
		14	2.20 V.	
		15	2.30 V.	
		16	2.40 V.	
		17	2.50 V.	
		18	2.60 V.	
		19	2.70 V.	
		20	2.806 V.	
		21	2.90 V.	
		22	3.00 V.	
		23	3.10 V.	
		24	3.20 V.	
		25	3.30 V.	
		26	3.30 V.	
		27	3.30 V.	
		28	3.30 V.	
		29	3.30 V.	
		30	3.30 V.	
		31	3.30 V.	
6:5	HYST		BoD Hysteresis control.	For device revision 0A: 0x3. For device revision 1B: 0x2
		0	25 mV.	
		1	50 mV.	
		2	75 mV.	
		3	100 mV.	
31:7	-		Reserved. Read value is undefined, only zero should be written.	undefined

13.4.7 Analog References fast wake-up Control register [Reset by: PoR]

Disclaimer: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

This shows Analog References fast wake-up Control register [Reset by: PoR].

Table 302. Analog References fast wake-up Control register [Reset by: PoR] (REFFASTWKUP, offset = 0x40)

Bit	Symbol	Value	Description	Reset value
0	LPWKUP		Analog References fast wake-up in case of wake-up from a low power mode (DEEP SLEEP, POWER DOWN and DEEP POWER DOWN):	0x1
		0b	Analog References fast wake-up feature is disabled in case of wake-up from any Low power mode.	
		1b	Analog References fast wake-up feature is enabled in case of wake-up from any Low power mode.	
1	HWWKUP		Analog References fast wake-up in case of Hardware Pin reset:	0x0
		0b	Analog References fast wake-up feature is disabled in case of Hardware Pin reset.	
		1b	Analog References fast wake-up feature is enabled in case of Hardware Pin reset.	
31-2	-		Reserved Read value is undefined, only zero should be written.	undefined.

13.4.8 32 KHz Crystal oscillator (XTAL) control register [Reset by: PoR, Brown Out Detectors Reset]

Disclaimer: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

This shows 32 KHz Crystal oscillator (XTAL) control register values [Reset by: PoR, Brown Out Detectors Reset].

Table 303. 32 KHz Crystal oscillator (XTAL) control register [Reset by: PoR, Brown Out Detectors Reset] (XTAL32K, offset = 0x4C)

Bit	Symbol	Value	Description	Reset value
0	-		Reserved Read value is undefined, only zero should be written.	undefined.
2-1	IREF		Reference output current selection inputs.	0x1
3	TEST		Oscillator Test Mode.	0x0
5-4	IBIAS		Bias current selection inputs.	0x1
7-6	AMPL		Oscillator amplitude selection inputs.	0x1
14-8	CAPBANKIN		Capa bank setting input.	0x40
21-15	CAPBANKOUT		Capa bank setting output.	0x40
22	CAPTESTSTARTSRCSEL		Source selection for xo32k_captest_start_ao_set.	0x0

Table 303. 32 KHz Crystal oscillator (XTAL) control register [Reset by: PoR, Brown Out Detectors Reset] (XTAL32K, offset = 0x4C) *…continued*

Bit	Symbol	Value	Description	Reset value
		0	Sourced from CAPTESTSTART.	
		1	Sourced from calibration.	
23	CAPTESTSTART		Start test.	0x0
24	CAPTESTENABLE		Enable signal for cap test.	0x0
25	CAPTESTOSCINSEL		Select the input for test.	0x0
		0	Oscillator output pin (osc_out).	
		1	Oscillator input pin (osc_in).	
31-26	-		Reserved Read value is undefined, only zero should be written.	undefined.

13.4.9 Analog comparator control register

Table 304. Analog comparator control register (Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset) (COMP, offset = 0x50)

Bit	Symbol	Value	Description	Reset value
0	-		Reserved. Read value is undefined, only zero should be written.	undefined.
1	HYST		Hysteris when hyst =’1’.	0x1
		0	Hysteresis is disable.	
		1	Hysteresis is enable.	
2	VREFINPUT		Dedicated control bit to select between VREF and VDDA (for the resistive ladder).	0x0
		0	Select internal VREF.	
		1	Select VDDA.	
3	LOWPOWER		Low power mode.	0x1
		0	High speed mode.	
		1	Low power mode (Low speed).	
6:4	PMUX		Control word for P multiplexer.	0x0
		0	VREF (See field VREFINPUT).	
		1	Pin P0_0.	
		2	Pin P0_9.	
		3	Pin P0_18.	
		4	Pin P1_14.	
		5	Pin P2_23.	

Table 304. Analog comparator control register (Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset) (COMP, offset = 0x50) *…continued*

Bit	Symbol	Value	Description	Reset value
9:7	NMUX		Control word for N multiplexer:	0x0
		0	VREF (See field VREFINPUT).	
		1	Pin P0_0.	
		2	Pin P0_9.	
		3	Pin P0_18.	
		4	Pin P1_14.	
		5	Pin P2_23.	
14:10	VREF		Control reference voltage step, per steps of (VREFINPUT/31).	0x0
15	-		Reserved. Read value is undefined, only zero should be written.	undefined
17:16	FILTERCGF_SAMPLEMODE		Control the filtering of the Analog Comparator output.	0x0
		0	Bypass mode. Filtering is disabled. The raw Analog Comparator output will be passed-through.	
		1	Filter 1 clock period. Any pulse duration shorter than one cycle of the designated filter clock (see FILTERCGF_CLKDIV) will be filtered-out. Pulse widths up to two cycles long may be filtered.	
		2	Filter 2 clock period. Any pulse duration shorter than two cycles of the designated filter clock will be filtered-out. Pulse widths up to three cycles long may be filtered.	
		3	Filter 3 clock period. Any pulse duration shorter than three cycles of the designated filter clock will be filtered-out. Pulse widths up to four cycles long may be filtered.	
20:18	FILTERCGF_CLKDIV		Filter clock divider. Filter clock equals the Analog Comparator clock divided by 2^FILTERCGF_CLKDIV.	0x0
		0	Filter clock period duration equals 1 Analog Comparator clock period.	
		1	Filter clock period duration equals 2 Analog Comparator clock period.	
		2	Filter clock period duration equals 4 Analog Comparator clock period.	
		3	Filter clock period duration equals 8 Analog Comparator clock period.	
		4	Filter clock period duration equals 16 Analog Comparator clock period.	
		5	Filter clock period duration equals 32 Analog Comparator clock period.	
		6	Filter clock period duration equals 64 Analog Comparator clock period.	

Table 304. Analog comparator control register (Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset) (COMP, offset = 0x50) *…continued*

Bit	Symbol	Value	Description	Reset value
		7	Filter clock period duration equals 128 Analog Comparator clock period.	
23:21	-		Reserved. Read value is undefined, only zero should be written.	undefined
31:24	-		Reserved. Read value is undefined, only zero should be written.	undefined

13.4.10 Wake-up I/O Control register

Disclaimer: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

The wake-up I/O control register is used to detect and control deep power down and wake up sources (reset by: PoR, Pin Reset, and Software Reset).

Table 305. Wake-up I/O Control register (WAKEUPIOCTRL, offset = 0x64) bit description

Bit	Symbol	Value	Description	Reset value
0	RISINGEDGEWAKEUP0		Enable / disable detection of rising edge events on Wake Up 0 pin in Deep Power Down modes.	0x0
		0b	Rising edge detection is disable.	
		1b	Rising edge detection is enable.	
1	FALLINGEDGE WAKEUP0		Enable / disable detection of falling edge events on Wake Up 0 pin in Deep Power Down modes.	0x0
		0b	Falling edge detection is disable.	
		1b	Falling edge detection is enable.	
2	RISINGEDGEWAKEUP1		Enable / disable detection of rising edge events on Wake Up 1 pin in Deep Power Down modes.	0x0
		0b	Rising edge detection is disable.	
		1b	Rising edge detection is enable.	
3	FALLINGEDGEWAKEUP1		Enable / disable detection of falling edge events on Wake Up 1 pin in Deep Power Down modes.	0x0
		0b	Falling edge detection is disable.	
		1b	Falling edge detection is enable.	
4	RISINGEDGEWAKEUP2		Enable / disable detection of rising edge events on Wake Up 2 pin in Deep Power Down modes.	0x0
		0b	Rising edge detection is disable.	
		1b	Rising edge detection is enable.	
5	FALLINGEDGEWAKEUP2		Enable / disable detection of falling edge events on Wake Up 2 pin in Deep Power Down modes.	0x0
		0b	Falling edge detection is disable.	
		1b	Falling edge detection is enable.	

Table 305. Wake-up I/O Control register (WAKEUPIOCTRL, offset = 0x64) *…continued*bit description

Bit	Symbol	Value	Description	Reset value
6	RISINGEDGEWAKEUP3		Enable / disable detection of rising edge events on Wake Up 3 pin in Deep Power Down modes.	0x0
		0b	Rising edge detection is disable.	
		1b	Rising edge detection is enable.	
7	FALLINGEDGEWAKEUP3		Enable / disable detection of falling edge events on Wake Up 3 pin in Deep Power Down modes.	0x0
		0b	Falling edge detection is disable.	
		1b	Falling edge detection is enable.	
8	MODEWAKEUP0		Configure wake up I/O 0 in Deep Power Down mode. To be used with MISCCTRL[8]:	0x0
		2'b00	No pull-up / no pull-down default at start-up.	
		2'b10	Pull-down.	
		2'b01	Pull-up.	
		2'b11	Reserved.	
9	MODEWAKEUP1		Configure wake up I/O 1 in Deep Power Down mode. To be used with MISCCTRL[9]:	0x0
		2'b00	No pull-up / no pull-down default at start-up.	
		2'b10	Pull-down.	
		2'b01	Pull-up.	
		2'b11	Reserved.	
10	MODEWAKEUP2		Configure wake up I/O 2 in Deep Power Down mode. To be used with MISCCTRL[10]:	0x0
		2'b00	No pull-up / no pull-down default at start-up.	
		2'b10	Pull-down.	
		2'b01	Pull-up.	
		2'b11	Reserved.	
11	MODEWAKEU P3		Configure wake up I/O 3 in Deep Power Down mode. To be used with MISCCTRL[11]:	0x0
		2'b00	No pull-up / no pull-down default at start-up.	
		2'b10	Pull-down.	
		2'b01	Pull-up.	
		2'b11	Reserved.	
31:12	-		Reserved. Read value is undefined, only zero should be written.	0x0

13.4.11 Wake-up I/O cause register

The wake-up I/O cause register allows to identify the wake-up I/O source from deep power-down mode.

Table 306. Wake-up I/O register (WAKEIOCAUSE, offset = 0x68)

Bit	Symbol	Value	Description	Reset value
0	WAKEUP0		Allows to identify Wake up I/O 0 as the wake-up source from deep-power down mode.	0x0
		0	Last wake up from deep-power down mode was NOT triggered by wake up I/O 0.	
		1	Last wake up from deep-power down mode was triggered by wake up I/O 0.	
1	WAKEUP1		Allows to identify Wake up I/O 1 as the wake-up source from deep-power down mode.	0x0
		0	Last wake up from deep-power down mode was NOT triggered by wake up I/O 1.	
		1	Last wake up from deep-power down mode was triggered by wake up I/O 1.	
2	WAKEUP2		Allows to identify Wake up I/O 2 as the wake-up source from deep-power down mode.	0x0
		0	Last wake up from deep-power down mode was NOT triggered by wake up I/O 2.	
		1	Last wake up from deep-power down mode was triggered by wake up I/O 2.	
3	WAKEUP3		Allows to identify Wake up I/O 3 as the wake-up source from deep-power down mode.	0x0
		0	Last wake up from deep-power down mode was NOT triggered by wake up I/O 3.	
		1	Last wake up from deep-power down mode was triggered by wake up I/O 3.	
31:4	-		Reserved. Read value is undefined, only zero should be written.	undefined

13.4.12 Status CLK register

Table 307. FRO and XTAL status register (Reset by: PoR, Brown Out Detectors reset) (STATUSCLK, offset = 0x74)

Bit	Symbol	Value	Description	Reset value
0	XTAL32KOK		XTAL oscillator 32 K OK signal when read as ‘1’. Not OK when read as ‘0’	0x0
1	-		Reserved. Read value is undefined, only zero should be written.	undefined
2	XTAL32KOSCFAILURE		XTAL32 kHz oscillator oscillation failure detection indicator.	0x1
		0	No oscillation failure has been detected since the last time this bit has been cleared.	
		1	At least one oscillation failure has been detected since the last time this bit has been cleared. Write '1' to clear.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

13.4.13 General purpose always on domain data storage

This register is managed and updated by the ROM Boot Code. It gathers some important System Status information like the last System reset cause and the number of fatal errors that occurred during the ROM boot. Though it is readable and writable, it can not be modified by any application.

General purpose always on domain data storage (Reset by: PoR, Brown Out Detectors Reset) (AOREG1, offset = 0x84)

Bit	Access	Symbol	Description	Reset value
3:0	RW	-	Reserved. Read value is undefined, only zero should be written.	0
4	RW	POR	The last chip reset was caused by a Power On Reset.	-
5	RW	PADRESET	The last chip reset was caused by a Pin Reset.	0
6	RW	BODRESET	The last chip reset was caused by a Brown Out Detector (BoD), either VBAT, BoD, or Core Logic BoD.	0
7	RW	SYSTEMRESET	The last chip reset was caused by a System Reset requested by the ARM CPU.	0
8	RW	WDTRESET	The last chip reset was caused by the Watchdog Timer.	0
9	RW	SWRRESET	The last chip reset was caused by a Software event.	0
10	RW	DPDRESET_WAKEUPIO	The last chip reset was caused by a Wake-up I/O reset event during a Deep Power-Down mode.	0
11	RW	DPDRESET_RTC	The last chip reset was caused by an RTC (either RTC Alarm or RTC wake up) reset event during a Deep Power-Down mode.	0
12	RW	DPDRESET_OSTIMER	The last chip reset was caused by an OS Event Timer reset event during a Deep Power-Down mode.	0
15:13	RW	-	Reserved.	-
19:16	RW	BOOTERRORCOUNTER	ROM Boot Fatal Error Counter.	0
31:20	RW	-	Reserved. Read value is undefined, only zero should be written.	-

13.4.14 Dummy Control bus to PMU [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset]

Disclaimer: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

This shows Dummy Control bus to PMU [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset].

Table 308. Dummy Control bus to PMU [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (MISCCTRL, offset = 0x90)

Bit	Symbol	Value	Description	Reset value
0	LDODEEPSLEEPREF		Select LDO Deep Sleep reference source.	0x0
		0b	LDO DEEP Sleep uses Flash buffer biasing as reference.	
		1b	LDO DEEP Sleep uses Band Gap 0.8V as reference.	
1	LDOMEMHIGHZMODE		Control the activation of LDO MEM High Z mode.	0x0

Table 308. Dummy Control bus to PMU [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (MISCCTRL, offset = 0x90) *…continued*

Bit	Symbol	Value	Description	Reset value
		0b	LDO MEM High Z mode is disabled.	
		1b	LDO MEM High Z mode is enabled.	
2	LOWPWR_FLASH_BUF			0x0
7-3	MISCCTRL_3_ 8		Reserved.	0x0
8	MODEWAKEUP0		Configure wake up I/O 0 in Deep Power Down mode To be used with WAKEUPIOCTRL[8]:	0x0
		2'b00	No pull-up / no pull-down default at start-up.	
		2'b10	Pull-down.	
		2'b01	Pull-up.	
		2'b11	Reserved.	
9	MODEWAKEUP1		Configure wake up I/O 1 in Deep Power Down mode To be used with WAKEUPIOCTRL[9]:	0x0
		2'b00	No pull-up / no pull-down default at start-up.	
		2'b10	Pull-down.	
		2'b01	Pull-up.	
		2'b11	Reserved.	
10	MODEWAKEUP2		Configure wake up I/O 2 in Deep Power Down mode. To be used with WAKEUPIOCTRL[10]:	0x0
		2'b00	No pull-up / no pull-down default at start-up.	
		2'b10	Pull-down.	
		2'b01	Pull-up.	
		2'b11	Reserved.	
11	MODEWAKEUP3		Configure wake up I/O 3 in Deep Power Down mode. To be used with WAKEUPIOCTRL[11]:	0x0
		2'b00	No pull-up / no pull-down default at start-up.	
		2'b10	Pull-down.	
		2'b01	Pull-up.	
		2'b11	Reserved.	
12	DISABLE_BLEED		Controls LDO MEM bleed current. This field is expected to be controlled by the Low Power Software only in DEEP SLEEP low power mode.	0x0
		0b	LDO_MEM bleed current is enabled.	
		1b	LDO_MEM bleed current is disabled. Should be set before entering in Deep Sleep low power mode and cleared after wake up from Deep Sleep low power mode.	
14-13	MISCCTRL_13 _14		Reserved.	0x0
15	WAKUPIO_RS T		WAKEUP IO event detector reset control.	0x0

Table 308. Dummy Control bus to PMU [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (MISCCTRL, offset = 0x90) *…continued*

Bit	Symbol	Value	Description	Reset value
		0b	Wakeup IO is not reset.	
		1b	Wakeup IO is reset.	
31-16	-		Reserved Read value is undefined, only zero should be written.	undefined.

13.4.15 RTC 1 kHz and 1 Hz clocks source control register

This register selects the source of the 32K clock to the whole system, including the RTC. It also controls the RTC clock dividers.

Table 309. RTC 1 kHz and 1 Hz clocks source control register (Reset by: PoR, Brown Out Detectors reset) (RTCOSC32K, offset = 0x98)

Bit	Symbol	Value	Description	Reset value
0	SEL		Selects either the XTAL32kHz or FRO32kHz as the 32K clock source for the whole system.	0x0
		0	FRO 32 kHz.	
		1	XTAL 32 kHz.	
3:1	CLK1kHzDIV		Actual division ratio is: 28 + CLK1 kHz	0x4
14:4	-		Reserved. Read value is undefined, only zero should be written.	undefined
15	CLK1kHzDIVUPDATEREQ		RTC 1 kHz clock divider status flag.	0x0
26:16	CLK1HZDIV		Actual division ratio is: 31744 + CLK1HZDIV.	0x3FF
29:27	-		Reserved. Read value is undefined, only zero should be written.	undefined
30	CLK1HZDIVHALT		Halts the divider counter.	0x0
31	CLK1HZDIVUPDATEREQ		RTC 1Hz divider status flag.	0x0

13.4.16 OS timer control register

Table 310. OS timer control register [Reset by: PoR, Brown Out Detectors Reset] (OSTIMER, offset = 0x9C)

Bit	Symbol	Value	Description	Reset value
0	SOFTRESET		Active high reset.	0x0
1	CLOCKENABLE		Enable OS event timer clock.	0x0
2	DPDWAKEUPENABLE		Wake up enable in deep-power down mode (To be used in enable deep-power down mode).	0x0
3	OSC32KPD		Power down oscillator 32 kHz (either FRO32 kHz or XTAL32 kHz according to RTCOSC32K.SEL).	0x1
		0	Running	
		1	Power-down	
31:4	-		Reserved. Read value is undefined, only zero should be written.	undefined

13.4.17 Power configuration register 0

The PDRUNCFG0 register controls the power to various analog blocks.

Table 311. Power configuration register 0 (PDRUNCFG0, offset = 0xB8) (Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset)

Bit	Symbol	Value	Description	Reset value
0	-		Reserved. Only zero should be written.	0x0
1	-		Reserved. Only zero should be written.	0x0
2	-		Reserved. Only zero should be written.	0x0
3	PDEN_BODVBAT		Controls power to VBAT Brown Out Detector (BOD).	0x0
		0	BOD VBAT is powered.	
		1	BOD VBAT is powered-down.	
4	-		Reserved. Read value is undefined, only zero should be written.	undefined
5	-		Reserved. Only zero should be written.	0x0
6	PDEN_FRO32K		Controls power to the Free Running Oscillator (FRO) 32 kHz. Remark: The 32 kHz Free Running Oscillator will be automatically enabled when: the RTC_OSP_PD bit in RTC control register = 0 and the SEL bit in RTCOSC32K register = 0 or when the OSC32KPD bit in OSTIMER register = 0 and the SEL bit in the RTCOSC32K = 0 or when the PDEN_FRO32K bit in PDRUNCFG0 = 0	0x1
		0	FRO32kHz is powered.	
		1	FRO32kHz is powered-down.	
7	PDEN_XTAL32K		Controls power to crystal 32 kHz.	0x1
		0	Crystal 32 kHz is powered.	
		1	Crystal 32 kHz is powered-down.	
8	PDEN_XTAL32M		Controls power to crystal 32 MHz.	0x1
		0	Crystal 32MHz is powered.	
		1	Crystal 32MHz is powered-down.	
9	PDEN_PLL0		Controls power to PLL0.	0x1
		0	PLL0 is powered.	
		1	PLL0 is powered-down.	
10	PDEN_PLL1		Controls power to PLL1.	0x1
		0	PLL1 is powered.	
		1	PLL1 is powered-down.	
11	PDEN_USBFSPHY [1]		Controls power to USB full speed PHY.	0x1
		0	USB full speed PHY is powered.	
		1	USB full speed PHY is powered-down.	
12	PDEN_USBHSPHY		Controls power to USB high speed PHY.	0x1
		0	USB HS PHY is powered.	
		1	USB HS PHY is powered-down.	
13	PDEN_COMP		Controls power to analog comparator.	0x1
		0	Analog comparator is powered.	
		1	Analog comparator is powered-down.	
15:14	-	-	Reserved. Read value is undefined, only zero should be written.	
16	-		Reserved. Only zero should be written.	0x0
17	-		Reserved. Only one should be written.	0x0

Table 311. Power configuration register 0 (PDRUNCFG0, offset = 0xB8) (Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset) *…continued*

Bit	Symbol	Value	Description	Reset value
18	PDEN_LDOUSBHS		Controls power to USB high speed LDO.	0x1
		0	USB high speed LDO is powered.	
		1	USB high speed LDO is powered-down.	
19	PDEN_AUXBIAS		Controls power to auxiliary biasing (AUXBIAS), and also provides power control to BIAS_VREF_1V.	0x1
		0	Auxiliary biasing is powered.	
		1	auxiliary biasing is powered-down.	
20	PDEN_LDOXO32M		Controls power to crystal 32 MHz LDO.	0x1
		0	Crystal 32 MHz LDO is powered.	
		1	Crystal 32 MHz LDO is powered-down.	
21	-		Reserved. Only one should be written.	0x0
22	PDEN_RNG		Controls power to all True Random Number Generator (TRNG) clock sources.	0x1
		0	TRNG clocks are powered.	
		1	TRNG clocks are powered-down.	
23	PDEN_PLL0_SSCG		Controls power to system PLL0 spread spectrum module.	0x1
		0	PLL0 spread spectrum module is powered.	
		1	PLL0 spread spectrum module is powered-down.	
24	-		Reserved. Read value is undefined, only zero should be written.	undefined
31:25	-		Reserved. Read value is undefined, only zero should be written.	undefined

[1] For VFBGA59 package parts, make sure PDEN_USBFSPHY bits are set to save power.

13.4.18 Power configuration set register 0

The power configuration set register 0 controls the power to various analog blocks.

Table 312. Power configuration set register 0 (PDRUNCFGSET0, offset = 0xC0) (Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset)

Bit	Symbol	Value	Description	Reset value
31:0	PDRUNCFGSET0		Writing ones to this register sets the corresponding bit or bits in the PDRUNCFG0 register, if they are implemented.	0x0

13.4.19 Power configuration clear register 0

The power configuration clear register 0 controls the power to various analog blocks.

Table 313. Power configuration clear register (PDRUNCFGCLR0, offset = 0xC8)(Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset)

Bit	Symbol	Value	Description	Reset value
31:0	PDRUNCFGCLR0		Writing ones to this register clears the corresponding bit or bits in the PDRUNCFG0 register, if they are implemented.	0x0

13.4.20 All SRAMs common control signals [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Software Reset]

Disclaimer: The values written to this register should not be changed. These registers are handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

This shows all SRAMs common control signals [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Software Reset].

Table 314. All SRAMs common control signals [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Software Reset] (SRAMCTRL, offset = 0xD4)

Bit	Symbol	Value	Description	Reset value
1-0	SMB		Source Biasing voltage.	0x1
		00b	Low leakage.	
		01b	Medium leakage.	
		10b	Highest leakage.	
		11b	Disable.	
4-2	RM		Read Margin control settings.	0x0
7-5	WM		Write Margin control settings.	0x0
8	WRME		Write read margin enable.	0x0
31-9	-		Reserved Read value is undefined, only zero should be written.	undefined.

14.1 How to read this chapter

The power profiles and power control APIs can be implemented using the power library from the SDK software package available on NXP.com.

14.2 Features

- Simple APIs to control power consumption and wake-up in all low power modes: sleep, deep-sleep, power-down, and deep power-down.
- Prepares the part to enter low power modes: deep-sleep, power-down and deep power-down.
- Manages power consumption for sleep and active modes.

14.3 General description

Control of device power consumption or entry to low power modes can be configured through simple calls to the low power profile API.

APIs exist to:

- Set up reduced power modes.
- Set up on-chip power based on the expected operating frequency.

Remark: Disable all interrupts before making calls to the power set voltage API call.

CPU0: Can be put in sleep mode. Only CPU0 can trigger deep-sleep, power-down, and deep power-down modes.

CPU1: Can be put in sleep mode. CPU1 can not trigger deep-sleep, power-down, and deep power-down modes. Also, CPU0 can be put in sleep mode while CPU1 is active.

14.4 Power related API descriptions

A Power API in the SDK power library is available to configure the system for expected performance requirements.

Table 315. Power API for active mode

Function prototype	API description	Section
`void POWER_SetVoltageForFreq(unit32_t system_freq_hz;`	Power API internal voltage configuration routine. This API configures the internal regulator for the desired active operating mode and frequency.	14.4.1

By default, the internal DC-DC converter output voltage is set to 1.10 volts for device revision 0A and 1.05 volts for device revision 1B to accommodate frequencies of 100 MHz and below. However, the voltage can be modified to accommodate higher frequency ranges as described in Table 316 “DC-DC converter output voltage settings”.

Table 316. DC-DC converter output voltage settings

Range	Frequency Ranges (MHz)	DC-DC Converter Output (V)
1	System Frequency ≤ 100	1.0 V - 1.075 V
2	System Frequency 101 to 130	1.025 V - 1.15 V
3	System Frequency 131 to 150	1.05 V - 1.2 V

The POWER_SetVoltageForFreq API call must always be used before initially setting the frequency and when changing the frequency from one range to another. The sequence of steps for switching from one frequency to another depends on whether the range is higher or lower.

When switching from a lower range to a higher range, the following series of steps must be followed:

1. Call the POWER_SetVoltageForFreq API call.
2. Call the SDK API CLOCK_SetFLASHAccessCyclesForFreq(uint32_t iFreq) function which will set up all the necessary flash timings (both the FMC and Flash Controller).
3. Use the SDK to update the system clock frequency to the new frequency.

When switching from a higher range to a lower range, the following series of steps must be followed:

1. Use the SDK to update the system clock frequency to the new frequency.
2. Call the SDK API CLOCK_SetFLASHAccessCyclesForFreq(uint32_t iFreq) function which will set up all flash timings (both the FMC and Flash Controller).
3. Call the POWER_SetVoltageForFreq API call.

Low power APIs provide functions to configure the system into the different low power modes: sleep, deep-sleep, power-down and deep power-down as shown in Table 317 “Low power API calls”.

Table 317. Low power API calls

Function prototype	API description	Section
`void POWER_EnterSleep(void);`	This API makes the CPU enter sleep mode.	14.4.2
`void POWER_EnterDeepSleep (uint32_t exclude_from_pd, uint32_t sram_retention_ctrl, uint64_t wakeup_interrupts, uint32_t hardware_wake_ctrl);`	This API configures the chip then enters deep-sleep mode.	14.4.3
`void POWER_EnterPowerDown(uint32_t exclude_from_pd, uint32_t sram_retention_ctrl, uint64_t wakeup_interrupts, uint32_t cpu_retention_ctrl);`	This API configures the chip then enters power-down mode.	14.4.4
`void POWER_EnterDeepPowerDown(uint32_t exclude_from_pd, uint32_t sram_retention_ctrl, uint64_t wakeup_interrupts, uint32_t wakeup_io_ctrl);`	This API configures the chip then enters deep power-down mode.	14.4.5

Remark:

- CPUs and System Clock frequency are switched to FRO12MHz and are NOT restored back by the POWER_EnterDeepSleep, POWER_EnterPowerDown, and POWER_EnterDeepPowerDownpower APIs.
- CPU0 interrupt enable registers (NVIC->ISER) are modified by POWER_EnterDeepSleep, POWER_EnterPowerDown, and POWER_EnterDeepPowerDown power APIs. They are restored in case of CPU retention (deep-sleep and in power-down) or if the low power mode is not entered (for example, a pending interrupt).
- The Non Maskable Interrupt (NMI) is disabled and its configuration before calling this function will be restored back in case of CPU retention – deep-sleep and in power-down - or if the low power mode is not entered (for instance because an interrupt is pending).
- The HARD FAULT handler should execute from SRAM and not from Flash. (The hard fault handler should initiate a full chip reset).

14.4.1 POWER_SetVoltageForFreq (unit32_t system_freq_hz)

This routine configures the device's internal power control settings according to the calling arguments. The goal is to prepare on-chip DC-DC converter to deliver the amount of power needed for the requested performance level, as defined by the CPU operating frequency.

Table 318. POWER_SetVoltageForFreq API routines

Routine	POWER_SetVoltageForFreq
SDK Prototype	`POWER_SetVoltageForFreq(uint32_t system_freq_hz);`
Input parameter	**Param0:** desired frequency (in Hz)
Description	Configures the internal device voltage in active mode.

14.4.1.1 Param0: frequency

The frequency is the clock rate the CPU will be using during the selected mode. This operand must represent an integer from 1 to 150000000 inclusive.

14.4.2 POWER_EnterSleep

This routine puts the device in sleep mode (The activity of the CPU - either CPU0 or CPU1 - that called this function is stopped).

Table 319. POWER_EnterSleep API routine

Routine	CHIPLOWPOWER_enter_ sleep
SKD Prototype	`void POWER_EnterSleep(void);`
Input parameter	None
Result	None
Description	-

implementation of POWER_EnterSleep.

```
void POWER_EnterSleep(void)
{
    uint32_t pmsk;
    pmsk = __get_PRIMASK(); /* Save interrupt configuration */
    __disable_irq(); /* Disable all interrupts */
    SCB->SCR &= ~SCB_SCR_SLEEPDEEP_Msk; /* processor uses sleep */
    __WFI(); /* Enter sleep mode */
    __set_PRIMASK(pmsk);   /* Restore interrupt configuration */
}
```

14.4.3 POWER_EnterDeepSleep

This routine prepares the part then enter "*deep-sleep*" low power mode. the API function configures which analog/digital components remain running, so that an interrupt from one of the analog/digital peripherals can wake up the part.

Table 320. POWER_EnterDeepSleep API routine

Routine	POWER_EnterDeepSleep
SKD Prototype	`void POWER_EnterDeepSleep (uint32_t exclude_from_pd, uint32_t sram_retention_ctrl, uint64_t wakeup_interrupts, uint32_t hardware_wake_ctrl);`
Input parameter	**Param0:** `exclude_from_pd` **Param1:** `sram_retention_ctrl` **Param2:** `wakeup_interrupts` **Param3:** `hardware_wake_ctrl`
Result	None
Description	Configure the deep-sleep low power mode: allows controlling which peripherals are powered up and which SRAM instances are in retention state in deep-sleep.

14.4.3.1 Param0: exclude_from_pd

The exclude_from_pd parameter defines which analog peripherals shall NOT be powered down and therefore can wake up the chip from deep-sleep. The excluded peripherals remain running in deep-sleep mode. For example, the FRO-1MHz oscillator must be running if the WWDT is to remain active in deep-sleep mode.

The exclude_from_pd parameter is a 32-bit value that corresponds to the definition of the table just below. For each bit field:

- '0': the module is powered down during deep-sleep.
- '1': the module is running during deep-sleep.

Table 321. Parameter exclude_from_pd

Bit	Symbol	Description	Value
0	-	Reserved	-
1	-	Reserved	-
2	BODCORE	Core logic brown out detector	0: Powered down 1: Running
3	BODVBAT	VBAT brown out detector	0: Powered down 1: Running
4	FRO1M	1 MHz free running oscillator	0: Powered down 1: Running
5	FRO192M	192 MHz free running oscillator	0: Powered down 1: Running
6	FRO32K	32 kHz free running oscillator	0: Powered down 1: Running
7	XTAL32K	32 kHz Crystal oscillator	0: Powered down 1: Running
8	XTAL32M	32 MHz Crystal oscillator	0: Powered down 1: Running
9	PLL0	1st general purpose PLL	0: Powered down 1: Running
10	PLL1	2nd general purpose PLL	0: Powered down 1: Running
11	USBFSPHY	USB full-speed physical	0: Powered down 1: Running
12	USBHSPHY	USB high-speed physical	0: Powered down 1: Running
13	COMP	Analog comparator	0: Powered down 1: Running
14	-	Reserved	-
15	GPADC	General purpose ADC	0: Powered down 1: Running
16	-	Reserved	-
17	-	Reserved	-
18	LDOUSBHS	USB high-speed regulator	0: Powered down 1: Running
19	AUXBIAS	ADC analog references	0: Powered down 1: Running
20	LDOXO32M	32 MHz Crystal oscillator regulator	0: Powered down 1: Running
21	LDOFLASHNV	Flash regulator	0: Powered down 1: Running

Table 321. Parameter exclude_from_pd *…continued*

Bit	Symbol	Description	Value
22	RNG	True random number generator	0: Powered down 1: Running
23	PLL0_SSCG	PLL0 spread spectrum clock generator	0: Powered down 1: Running
24	-	Reserved	-
31:25	-	Reserved	-

14.4.3.2 Param1: sram_retention_ctrl

The sram_retention_ctrl parameter defines which SRAM instances will be put in “retention” mode during deep-sleep. SRAM instances in r*etention mode* do not lose their content but they cannot be involved in a DMA transfer during deep-sleep. SRAM instances that are not required to be put in *Retention mode* during deep-sleep will keep the state they had before calling the API, meaning:

- If the SRAM instance was in *Active mode*, it will stay in *Active mode* during deep-sleep and after wake up from deep-sleep. Such an SRAM instance can be involved in DMA transfer during deep-sleep.
- If the SRAM instance was in *Shutdown mode*, it will stay in *Shutdown mode* during deep-sleep and after wake up from deep-sleep.

The sram_retention_ctrl parameter is a 32-bit value that corresponds to the definition of the table just below. For each bit field:

- ‘0’: during deep-sleep, the SRAM instance keeps the state it has before entering deep-sleep.
- ‘1’: the SRAM instance will be put in *Retention mode* during deep-sleep.

Table 322. Parameter sram_retention_ctrl

Bit	SRAM instance	Value
0	**RAM_X0** (16 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
1	**RAM_X1** (8 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
2	**RAM_X2** (4 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
3	**RAM_X3** (4 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
4	**RAM_00** (32 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
5	**RAM_01** (32 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
6	**RAM_10** (64 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
7	**RAM_20** (64 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.

Table 322. Parameter sram_retention_ctrl *…continued*

Bit	SRAM instance	Value
8	**RAM_30** (32 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
9	**RAM_31** (32 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
10	**RAM_40** (4 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
11	**RAM_41** (4 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
12	**RAM_42** (4 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
13	**RAM_43** (4 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
14	**RAM_USB** (16 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
31:15	Reserved	-

14.4.3.3 Param2: wakeup_interrupts

The wakeup_interrupts parameter defines which peripheral interrupts can be a wake-up source during deep-sleep.

The table below describes, for each low power mode, if an interrupt can be the source for a wake-up.

Table 323. Parameter wakeup_interrupts

Bit	Wake-up source	Description	Deep-sleep	Power-down	Deep power-down
0	WAKEUP_SYS	Watchdog timer, BoDs	YES	NO	NO
1	WAKEUP_SDMA0	System DMA	YES	NO	NO
2	WAKEUP_GPIO_GLOBALINT0	GINT0	YES	YES	NO
3	WAKEUP_GPIO_GLOBALINT1	GINT1	YES	YES	NO
4	WAKEUP_GPIO_INT0_0	GPIO	YES	NO	NO
5	WAKEUP_GPIO_INT0_1	GPIO	YES	NO	NO
6	WAKEUP_GPIO_INT0_2	GPIO	YES	NO	NO
7	WAKEUP_GPIO_INT0_3	GPIO	YES	NO	NO
8	WAKEUP_UTICK	Micro-Tick timer	YES	NO	NO
9	WAKEUP_MRT	Multi rate timer	NO	NO	NO
10	WAKEUP_CTIMER0	Standard Counter/Timer 0	YES	NO	NO
11	WAKEUP_CTIMER1	Standard Counter/Timer 1	YES	NO	NO
12	WAKEUP_SCT	SCTimer/PWM	NO	NO	NO
13	WAKEUP_CTIMER3	Standard Counter/Timer 3	YES	NO	NO
14	WAKEUP_FLEXCOMM0	USART, SPI, I^2C, I^2S	YES	NO	NO
15	WAKEUP_FLEXCOMM1	USART, SPI, I^2C, I^2S	YES	NO	NO
16	WAKEUP_FLEXCOMM2	USART, SPI, I^2C, I^2S	YES	NO	NO

Table 323. Parameter wakeup_interrupts *…continued*

Bit	Wake-up source	Description	Deep-sleep	Power-down	Deep power-down
17	WAKEUP_FLEXCOMM3	USART, SPI, I^2C, I^2S	YES	YES	NO
18	WAKEUP_FLEXCOMM4	USART, SPI, I^2C, I^2S	YES	NO	NO
19	WAKEUP_FLEXCOMM5	USART, SPI, I^2C, I^2S	YES	NO	NO
20	WAKEUP_FLEXCOMM6	USART, SPI, I^2C, I^2S	YES	NO	NO
21	WAKEUP_FLEXCOMM7	USART, SPI, I^2C, I^2S	YES	NO	NO
22	WAKEUP_ADC	General purpose ADC	NO	NO	NO
23	-	-	-	-	-
24	WAKEUP_ACMP	Analog comparator	YES	YES	NO
25	-	-	-	-	-
26	-	-	-	-	-
27	WAKEUP_USB0_NEEDCLK	USB full-speed	YES	NO	NO
28	WAKEUP_USB0	USB full-speed	YES	NO	NO
29	WAKEUP_RTC_LITE_ALARM_WAKEUP	RTC	YES	YES	YES
30	-	-	-	-	-
31	WAKEUP_WAKEUP	Wakeup for low power mode (Power Down) use when wakeupio is enabled during power down. Bit 63 needs to be enabled as well.	NO	YES	NO
32	WAKEUP_GPIO_INT0_4	GPIO	YES	NO	NO
33	WAKEUP_GPIO_INT0_5	GPIO	YES	NO	NO
34	WAKEUP_GPIO_INT0_6	GPIO	YES	NO	NO
35	WAKEUP_GPIO_INT0_7	GPIO	YES	NO	NO
36	WAKEUP_CTIMER2	Standard Counter/Timer 2	YES	NO	NO
37	WAKEUP_CTIMER4	Standard Counter/Timer 4	YES	NO	NO
38	WAKEUP_OS_EVENT_TIMER	OS event timer	YES	YES	YES
39	-	-	-	-	-
40	-	-	-	-	-
41	-	-	-	-	-
42	WAKEUP_SDIO	SDIO controller interrupt	NO	NO	NO
43	-	-	-	-	-
44	-	-	-	-	-
45	-	-	-	-	-
46	-	-	-	-	-
47	WAKEUP_USB1	USB high-speed	YES	NO	NO
48	WAKEUP_USB1_NEEDCLK	USB high-speed	YES	NO	NO
49	WAKEUP_SEC_HYPERVISOR_CALL	Hypervisor security violation	NO	NO	NO
50	WAKEUP_SEC_GPIO_INT0_0	Secure GPIO	YES	NO	NO
51	WAKEUP_SEC_GPIO_INT0_1	Secure GPIO	YES	NO	NO
52	WAKEUP_PLU	Programmable logic	YES	NO	NO
53	WAKEUP_SEC_VIO	Security violation	NO	NO	NO

Table 323. Parameter wakeup_interrupts *…continued*

Bit	Wake-up source	Description	Deep-sleep	Power-down	Deep power-down
54	WAKEUP_SHA	HASH-AES256	NO	NO	NO
55	WAKEUP_CASPER	CASPER	NO	NO	NO
56	WAKEUP_PUF	Physical unclonable function	NO	NO	NO
57	WAKEUP_PQ	Power quad	NO	NO	NO
58	WAKEUP_SDMA1	Secure system DMA	YES	NO	NO
59	WAKEUP_HS_SPI	high-speed SPI	YES	NO	NO
62:61	-	-	-	-	-
63	WAKEUP_ALLWAKEUPIOS	Wakeup pins. Bit 31 needs to be enabled as well.	NO	YES	NO

The wakeup_interrupts parameter is a 64-bit value. For each bit field:

- '0': the associated peripheral cannot be a wake up source during deep-sleep.
- '1': the associated peripheral can be a wake up source during deep-sleep.

14.4.3.4 Param3: hardware_wake_ctrl

The primary goal of the hardware_wake_ctrl parameter is to provide the possibility for all Flexcomm Interfaces and the high-speed SPI to have DMA service during deep-sleep without waking up entire device.

These wake-ups are based on Flexcomm Interfaces and high-speed SPI peripherals FIFO levels.

Table 324. Parameter hardware_wake_ctrl

Bit	Symbol	Description	Value
0	Reserved		Shall always be set to '0'
1	HWWAKE_PERIPHERALS	Wake for Flexcomms. Any Flexcomm FIFO reaching the level specified by its own TXLVL will cause peripheral clocking to wake up temporarily while the related status is asserted.	0: Disabled 1: Enabled
2	Reserved	-	-
3	HWWAKE_SDMA0	Wake for DMA0. DMA0 being busy will cause peripheral clocking to remain running until DMA completes. Used in conjunction with HWWAKE_PERIPHERALS.	0: Disabled 1: Enabled
4	Reserved	Should always be set to zero.	Shall always be set to "0"
5	HWWAKE_SDMA1	Wake for DMA1. DMA0 being busy will cause peripheral clocking to remain running until DMA completes. Used in conjunction with HWWAKE_PERIPHERALS.	0: Disabled 1: Enabled
30:6	Reserved	Should always be set to zero.	Shall always be set to "0"
31	Reserved	Should always be set to zero.	Shall always be set to "0"

14.4.4 POWER_EnterPowerDown

This routine prepares the part then enter *power-down* low power mode. the API function configures which analog/digital components remain running, so that an interrupt from one of the analog/digital peripherals can wake up the part.

Table 325. POWER_EnterPowerDown API routine

Routine	POWER_EnterPowerDown
SKD Prototype	`void POWER_EnterPowerDown (uint32_t exclude_from_pd, uint32_t sram_retention_ctrl, uint64_t wakeup_interrupts, uint32_t cpu_retention_ctrl);`
Input parameter	**Param0:** `exclude_from_pd` **Param1:** `sram_retention_ctrl` **Param2:** `wakeup_interrupts` **Param3:** `cpu_retention_ctrl`
Result	None
Description	Configure the power-down low power mode: allows controlling which peripherals are powered up and which SRAM instances are in retention state in power-down.

Remark:

- It is the responsibility of the user to make sure that SRAM instance containing the stack used to call this software function WILL BE preserved during low power via parameter *sram_retention_ctrl*.

14.4.4.1 Param0: exclude_from_pd

The exclude_from_pd parameter defines which analog peripherals shall NOT be powered down and therefore can wake up the chip from power-down.

The exclude_from_pd parameter is a 32-bit value that corresponds to the definition of the table just below. For each bit field:

- '0': the module is powered down during power-down.
- '1': the module is running during power-down.

Table 326. Parameter exclude_from_pd

Bit	Symbol	Description	Value
0	-	Reserved	-
1	BIAS	Analog references	0: Powered down 1: Running
5:2	-	Reserved	-
6	FRO32K	32 kHz free running oscillator	0: Powered down 1: Running
7	XTAL32K	32 kHz Crystal oscillator	0: Powered down 1: Running
12:8	-	Reserved	-
13	COMP	Analog comparator	0: Powered down 1: Running
31:14	-	Reserved	-

Only the FRO 32kHz, the crystal 32 kHz, the analog comparator, the analog references (BIAS) and the memories regulator (LDOMEM) can be kept running in power-down mode.

The analog references (BIAS) are required only when the analog comparator is a wake-up source.

14.4.4.2 Param1: sram_retention_ctrl

The sram_retention_ctrl parameter defines which SRAM instances will be put in *Retention* mode during power-down. SRAM instances in *Retention mode* do not lose their content. SRAM instances that are not required to be put in *Retention mode* during power-down will be shut down (meaning their content will be lost upon wake-up from power-down).

The sram_retention_ctrl parameter is a 32-bit value that corresponds to the definition of the table just below. For each bit field:

- '0': during power-down, the SRAM instance loses its content.
- '1': the SRAM instance will be put in *Retention mode* during power-down.

Note: Address range [0x0400_6000 - 0x0400_65FF] inside RAMX_2 is used to store the state of CPU0 (which means that any user data in this area prior to calling the low power API will be lost). Therefore, RAM_X2 retention mode should always be enabled during power-down mode. If the user application uses power-down mode then it is recommended to configure SRAM_X2 to secure-privilege level.

Table 327. Parameter sram_retention_ctrl

Bit	SRAM instance	Value
0	**RAM_X0** (16 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
1	**RAM_X1** (8 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down 1: SRAM in retention mode during power-down/deep power-down.
2	**RAM_X2** (4 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
3	**RAM_X3** (4 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
4	**RAM_00** (32 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
5	**RAM_01** (32 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
6	**RAM_10** (64 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
7	**RAM_20** (64 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
8	**RAM_30** (32 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
9	**RAM_31** (32 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
10	**RAM_40** (4 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
11	**RAM_41** (4 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.

Table 327. Parameter sram_retention_ctrl

Bit	SRAM instance	Value
12	**RAM_42** (4 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
13	**RAM_43** (4 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
14	**RAM_USB** (16 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
31:15	Reserved	Shall always be written with 0x0.

14.4.4.3 Param2: wakeup_interrupts

The wakeup_interrupts parameter defines which peripheral interrupts can be a wake-up source during power-down.

See The table (reference table in deep-sleep param2 description) to see which interrupt can be a wake-up source during power-down.

The wakeup_interrupts parameter is a 64-bit value. For each bit field:

- '0': the associated peripheral cannot be a wake up source during power-down.
- '1': the associated peripheral can be a wake up source during power-down.

14.4.4.4 Param3: cpu_retention_ctrl

In power-down mode, the CPU0 state is retained (cpu_retention_ctrl must be set to 0x1).

CPU0 state retention is implemented by shifting the CPU0 registers values inside SRAM instance RAMX_2, meaning that RAMX_2 must be kept in retention, see Section 14.4.3.2 "Param1: sram_retention_ctrl". Along with CPU0, the state of AHB security controller and PRINCE registers values will also be shifted in RAMX_2. Address range [0x0400_6000 - 0x0400_65FF] inside RAMX_2 is used, which means that any data in this area prior to calling the low power API will be lost.

After a wake-up event occurs, CPU0 will resume code execution after the call to the low power API function.

When CPU0 state is retained, all SRAM instances that contain CPU0 variables (stack and heap) must also be retained, see Section 14.4.3.2 "Param1: sram_retention_ctrl".

The cpu_retention_ctrl parameter is a 32-bit value defined below:

Table 328. Parameter cpu_retention_ctrl

Bit	Symbol	Description	Value
0	CPU_RETENTION	Control CPU0 retention in power-down mode. PRINCE, and AHB security controller states will also be retained.	Must be set to 1.
31:1	Reserved	-	Shall always be written with 0x0.

14.4.5 POWER_EnterDeepPowerDown

This routine prepares the part then enter "deep power-down" low power mode. the API function configures which analog/digital components remain running, so that an interrupt from one of the analog/digital peripherals can wake up the part.

Table 329. POWER_EnterDeepPowerDown API routine

Routine	POWER_EnterDeepPowerDown
SKD Prototype	`void POWER_EnterDeepPowerDown (uint32_t exclude_from_pd, uint32_t sram_retention_ctrl, uint64_t wakeup_interrupts, uint32_t wakeup_io_ctrl);`
Input parameter	**Param0:** `exclude_from_pd` **Param1:** `sram_retention_ctrl` **Param2:** `wakeup_interrupts` **Param3:** `wakeup_io_ctrl`
Result	None
Description	Configure the deep power-down low power mode: allows controlling which peripherals are powered up and which SRAM instances are in retention state in deep power-down.

14.4.5.1 Param0: exclude_from_pd

The exclude_from_pd parameter defines which analog peripherals shall NOT be powered down and therefore can wake up the chip from power-down.

The exclude_from_pd parameter is a 32-bit value that corresponds to the definition of the table just below. For each bit field:

- '0': the module is powered down during deep-sleep.
- '1': the module is running during deep-sleep.

Table 330. Parameter exclude_from_pd

Bit	Symbol	Description	Value
5:0	-	Reserved.	-
6	FRO32K	32 kHz free running oscillator.	0: Powered down 1: Running
7	XTAL32K	32 kHz Crystal oscillator.	0: Powered down 1: Running.
31:8	-	Reserved.	-

Only the FRO 32kHz, and the Crystal 32 kHz can be kept running in deep power-down mode.

14.4.5.2 Param1: sram_retention_ctrl

The sram_retention_ctrl parameter defines which SRAM instances will be put in *Retention* mode during deep power-down. SRAM instances in *Retention mode* do not lose their content. SRAM instances that are not required to be put in *Retention mode* during deep power-down will be shut down (meaning their content will be lost upon wake-up from deep power-down.

The sram_retention_ctrl parameter is a 32-bit value that corresponds to the definition of the table just below. For each bit field:

- '0': during deep power-down, the SRAM instance loses its content.
- '1': the SRAM instance will be put in *Retention mode* during deep power-down.

Note: RAM_X0 (16 KB), RAMX1 (8 KB) and RAM_00 (32 KB) cannot be retained during deep power-down because they are used by the Boot ROM when waking-up from a deep power-down. As a consequence, the maximum amount of SRAM that can be retained during deep power-down is 264 KB).

Table 331. Parameter sram_retention_ctrl

Bit	SRAM instance	Value
0	**Reserved**	Only write 0x0
1	**Reserved**	Only write 0x0
2	**RAM_X2** (4 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
3	**RAM_X3** (4 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
4	**Reserved**	Only write 0x0
5	**RAM_01** (32 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
6	**RAM_10** (64 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
7	**RAM_20** (64 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
8	**RAM_30** (32 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
9	**RAM_31** (32 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
10	**RAM_40** (4 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
11	**RAM_41** (4 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
12	**RAM_42** (4 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
13	**RAM_43** (4 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
14	**RAM_USB** (16 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
31:15	Reserved	Only write 0x0

14.4.5.3 Param2: wakeup_interrupts

The wakeup_interrupts parameter defines which peripheral interrupts can be a wake-up source during deep power-down(see table 296).

Only WAKEUP_RTC_LITE_ALARM_WAKEUP and WAKEUP_OS_EVENT_TIMER interrupt can wake-up the part from a deep power-down.

The wakeup_interrupts parameter is a 64-bit value. Only bits 29 (WAKEUP_RTC_LITE_ALARM_WAKEUP) and 38 (WAKEUP_OS_EVENT_TIMER) are meaningful. All others bits are ignored. For each meaningful bit field:

- ‘0’: the associated peripheral cannot be a wake up source during deep power-down.
- ‘1’: the associated peripheral can be a wake up source during deep power-down.

14.4.5.4 Param3: wakeup_io_ctrl

The wake_up_io_ctrl parameter allows to configure the four wake-up pins that can wake-up the part from deep power-down mode.

Table 332 shows wake_up_io_ctrl parameter is a 32-bit value

Table 332. Parameter wakeup_io_ctrl

Bit	Symbol	Description	Value
0	RISINGEDGEWAKEUP0	Enable / disable detection of rising edge events on wake-up pin 0 in deep power-down modes.	0: Rising edge detection is disable 1: Rising edge detection is enable
1	FALLINGEDGEWAKEUP0	Enable / disable detection of falling edge events on wake-up pin 0 in deep power-down modes.	0: Falling edge detection is disable 1: Falling edge detection is enable
2	RISINGEDGEWAKEUP1	Enable / disable detection of rising edge events on wake-up pin 1 in deep power-down modes.	0: Rising edge detection is disable 1: Rising edge detection is enable
3	FALLINGEDGEWAKEUP1	Enable / disable detection of falling edge events on wake-up pin 1 in deep power-down modes.	0: Falling edge detection is disable 1: Falling edge detection is enable
4	RISINGEDGEWAKEUP2	Enable / disable detection of rising edge events on wake-up pin 2 in deep power-down modes.	0: Rising edge detection is disable 1: Rising edge detection is enable
5	FALLINGEDGEWAKEUP2	Enable / disable detection of falling edge events on wake-up pin 2 in deep power-down modes.	0: Falling edge detection is disable 1: Falling edge detection is enable
6	RISINGEDGEWAKEUP3	Enable / disable detection of rising edge events on wake-up pin 3 in deep power-down modes.	0: Rising edge detection is disable 1: Rising edge detection is enable
7	FALLINGEDGEWAKEUP3	Enable / disable detection of falling edge events on wake-up pin 3 in deep power-down modes.	0: Falling edge detection is disable 1: Falling edge detection is enable
8	PULLUPDOWNWAKEUP0	Enable Pull-down or Pull-up for wake-up pin 0 in deep power-down modes. This bit field is used only when: Wake-up pin 0 is disabled (indicated by both RISINGEDGEWAKEUP0=0 and FALLINGEDGEWAKEUP0=0) and the activation of the wake-up pin 0 pull-up or pull-down is enabled (DISABLEPULLUPDOWNWAKEUP0=0).	0: Pull-down 1: Pull-up
9	PULLUPDOWNWAKEUP1	Enable Pull-down or Pull-up for wake-up pin 1 in deep power-down modes. This bit field is used only when: Wake-up pin 1 is disabled (indicated by both RISINGEDGEWAKEUP1=0 and FALLINGEDGEWAKEUP1=0) and the activation of the wake-up pin 1 pull-up or pull-down is enabled (DISABLEPULLUPDOWNWAKEUP1=0).	0: Pull-down 1: Pull-up

Table 332. Parameter wakeup_io_ctrl

Bit	Symbol	Description	Value
10	PULLUPDOWNWAKEUP2	Enable Pull-down or Pull-up for wake-up pin 2 in deep power-down modes. This bit field is used only when: Wake-up pin 2 is disabled (indicated by both RISINGEDGEWAKEUP2=0 and FALLINGEDGEWAKEUP2=0) and the activation of the wake-up pin 2 pull-up or pull-down is enabled (DISABLEPULLUPDOWNWAKEUP2=0).	0: Pull-down 1: Pull-up
11	PULLUPDOWNWAKEUP3	Enable Pull-down or Pull-up for wake-up pin 3 in deep power-down modes. This bit field is used only when: Wake-up pin 3 is disabled (indicated by both RISINGEDGEWAKEUP3=0 and FALLINGEDGEWAKEUP3=0) and the activation of the wake-up pin 3 pull-up or pull-down is enabled (DISABLEPULLUPDOWNWAKEUP3=0).	0: Pull-down 1: Pull-up
12	DISABLEPULLUPDOWNWAKEUP0	Controls wake-up pin 0 pull-up/pull-down (see PULLUPDOWNWAKEUP0).	0: Pull-up/Pull-down is enabled. 1: Pull-up/Pull-down is disabled.
13	DISABLEPULLUPDOWNWAKEUP1	Controls wake-up pin 1 pull-up/pull-down (see PULLUPDOWNWAKEUP1).	0: Pull-up/Pull-down is enabled. 1: Pull-up/Pull-down is disabled.
14	DISABLEPULLUPDOWNWAKEUP2	Controls wake-up pin 2 pull-up/pull-down (see PULLUPDOWNWAKEUP2).	0: Pull-up/Pull-down is enabled. 1: Pull-up/Pull-down is disabled.
15	DISABLEPULLUPDOWNWAKEUP3	Controls wake-up pin 3 pull-up/pull-down (see PULLUPDOWNWAKEUP3).	0: Pull-up/Pull-down is enabled. 1: Pull-up/Pull-down is disabled.
16	USEEXTERNALPULLUPDOWN0	Enable usage of the external pull-up/down for Wake-up pin 0	0: use internal Pull-up/Pull-down. 1: use external Pull-up/Pull-down.
17	USEEXTERNALPULLUPDOWN1	Enable usage of the external pull-up/down for Wake-up pin 1	0: use internal Pull-up/Pull-down. 1: use external Pull-up/Pull-down.
18	USEEXTERNALPULLUPDOWN2	Enable usage of the external pull-up/down for Wake-up pin 2	0: use internal Pull-up/Pull-down. 1: use external Pull-up/Pull-down.
19	USEEXTERNALPULLUPDOWN3	Enable usage of the external pull-up/down for Wake-up pin 3	0: use internal Pull-up/Pull-down. 1: use external Pull-up/Pull-down.
31:20	-	Reserved. Must be set to 0x0	0x0

14.5 Functional Description

14.5.1 Enter deep-sleep mode

The four variables below are used in all subsequent examples:

```
Uint32_t exclude_from_pd;  /* */
Uint32_t sram_retention_ctrl; /* */
Uint32_t wakeup_interrupts; /* */
Uint32_t hardware_wake_ctrl ; /* */
```

14.5.1.1 Enter deep-sleep mode with wake up by RTC, using FRO32kHz as clock source, all SRAM instances in retention mode.

```
/*
   * - Configure RTC and FRO32kHz first, then call the sequence below
*/
exclude_from_pd = LOWPOWER_PDCTRL0_PDEN_FRO32K; /* The RTC will use the FRO 32 kHz as
     clock source */
sram_retention_ctrl = 0x7FFF; /* All RAM instances will be retained */
wakeup_interrupts = WAKEUP_RTC_LITE_ALARM_WAKEUP; /* RTC */
hardware_wake_ctrl  = 0; /* No DMA transfer during deep-sleep */
/* Enter Deep-sleep mode */
POWER_EnterDeepSleep (exclude_from_pd,sram_retention_ctrl, wakeup_interrupts,
     hardware_wake_ctrl);
```

14.5.1.2 Enter deep-sleep mode with wake-up by system DMA0

```
/*
   * - Configure any flexcomm in SPI receiver mode, and System DMA0 such that data
     received during deep-sleep on SPI will be transferred to RAM_X2 by System DMA0; a
     wake-up event will be fired when the required number of data transfered by DMA0
     is reached; then call the sequence below
*/
exclude_from_pd = 0; /* All analog peripherals shutdown */
sram_retention_ctrl = 0x7FFF & (~LOWPOWER_SRAMRETCTRL_RETEN_RAMX2); /* All RAM
     instances will be retained, except RAM_X2 RAM instance which will be kept in
     Active state because it is involved in DMA transfer during deep-sleep */
wakeup_interrupts = WAKEUP_SDMA0; /* System DMA0 */
hardware_wake_ctrl  = LOWPOWER_HWWAKE_SDMA0 | LOWPOWER_HWWAKE_PERIPHERALS; /* Allow
     DMA transfer without leaving deep-sleep mode */
/* Enter deep-sleep mode */
POWER_EnterDeepSleep (exclude_from_pd,sram_retention_ctrl, wakeup_interrupts,
     hardware_wake_ctrl);
```

14.5.2 Enter power-down mode

The following four variables are used in all subsequent examples:

```
Uint32_t sram_retention_ctrl; /* */
Uint32_t wakeup_interrupts; /* */
Uint32_t cpu_retention_ctrl ; /* */
```

14.5.2.1 Enter power-down mode with wake up by RTC, using FRO32kHz as clock source, CPU state retained, content of RAM_X2 and RAM_X3 retained

```
/*
 * - Configure RTC and FRO32kHz first, then call the sequence below
   */
   exclude_from_pd = LOWPOWER_PDCTRL0_PDEN_FRO32K; /* The RTC will use the FRO 32 kHz
      as clock source */
 sram_retention_ctrl = LOWPOWER_SRAMRETCTRL_RETEN_RAMX2 |
      LOWPOWER_SRAMRETCTRL_RETEN_RAMX3; /* RAM instances RAM_X2 & RAM_X3 content will
      be retained */
 wakeup_interrupts = WAKEUP_RTC_LITE_ALARM_WAKEUP; /* RTC */
 cpu_retention_ctrl = 1; /* CPU state retention enabled */
/* Enter power-down mode */
POWER_EnterDeepPowerDown (exclude_from_pd,sram_retention_ctrl, wakeup_interrupts,
      cpu_retention_ctrl );
```

14.5.2.2 Enter power-down mode with wake up by any GPIO in Port0 and Port1, CPU state retained, all SRAM instances retained

```
/*
 * - Configure Group GPIO input module 0/1 with the desired GPIO as wake up source,
      then call the sequence below
   */
   exclude_from_pd = 0 */
 sram_retention_ctrl = 0x7FFF; /* All RAM instances retained */
 wakeup_interrupts = WAKEUP_GPIO_GLOBALINT0 | WAKEUP_GPIO_GLOBALINT1; /* Group GPIO
      input module 0/1 */
 cpu_retention_ctrl = 1; /* CPU state retention enabled */
/* Enter Power-down mode */
CHIPLOWPOWER_enter_ powerdown (exclude_from_pd,sram_retention_ctrl,
      wakeup_interrupts, cpu_retention_ctrl);
```

14.5.2.3 Enter power-down mode with wake-up by Flexcomm3 (SPI or I²C), CPU state retained

```
 /*
      * - Configure the Flexcomm3 as SPI or I²C, in receiver mode, then call the
      sequence below
   */
exclude_from_pd = 0;
sram_retention_ctrl = LOWPOWER_SRAMRETCTRL_RETEN_RAMX2 |
LOWPOWER_SRAMRETCTRL_RETEN_RAMX3; /* RAM instances RAM_X2 & RAM_X3 content will
be retained, because they contain CPU stacks and variables for instance */
      LOWPOWER_SRAMRETCTRL_RETEN_RAMX3; /* RAM instances RAM_X2 & RAM_X3 content will
      be retained, because they contain CPU stacks and variables for instance */
wakeup_interrupts = WAKEUP_FLEXCOMM3; /* Flexcomm 3 */
cpu_retention_ctrl = 1; /* CPU state retention enabled */
/* Enter Power-down mode */
CHIPLOWPOWER_enter_ powerdown (exclude_from_pd,sram_retention_ctrl,
      wakeup_interrupts, cpu_retention_ctrl);
```

Note: In case UART is used as wake-up source in Flexcomm3, a 32-kHz clock source need to be enabled inside the IC. The unique baudrate supported is 9600 Baud.

14.5.2.4 Enter power-down mode with wake-up by analog comparator

```
 /*
   * - Configure the Analog Comparator, then call the sequence below
   */
   exclude_from_pd = LOWPOWER_SRAMRETCTRL_RETEN_RAMX2 |
LOWPOWER_SRAMRETCTRL_RETEN_RAMX3; /* RAM instances RAM_X2 & RAM_X3 content will
be retained, because they contain CPU stacks and variables for instance */ Analog
     References (BIAS) are required when Analog Comparator is turned on during
     power-down */
 sram_retention_ctrl = LOWPOWER_SRAMRETCTRL_RETEN_RAMX2 |
LOWPOWER_SRAMRETCTRL_RETEN_RAMX3; /
 wakeup_interrupts = WAKEUP_ACMP; /* Analog Comparator */
 cpu_retention_ctrl = 1; /* CPU state retention enabled */
/* Enter Power-down mode */
CHIPLOWPOWER_enter_ powerdown (exclude_from_pd,sram_retention_ctrl,
     wakeup_interrupts, cpu_retention_ctrl);
```

14.5.2.5 Enter deep power-down mode

The following four variables are used in all subsequent examples:

```
Uint32_t exclude_from_pd;  /* */

   Uint32_t sram_retention_ctrl; /* */

Uint32_t wakeup_interrupts; /* */

Uint32_t wakeup_io_ctrl ; /* */
```

14.5.2.6 Enter deep power-down mode with wake-up by RTC, using FRO32kHz as clock source, content of RAM_X2 and RAM_X3 retained

```
 /*
   * - Configure RTC and FRO32kHz first, then call the sequence below
   */
   exclude_from_pd = LOWPOWER_PDCTRL0_PDEN_FRO32K | LOWPOWER_PDCTRL0_PDEN_LDOMEM; /*
     The RTC will use the FRO 32 kHz as clock source */
 sram_retention_ctrl = LOWPOWER_SRAMRETCTRL_RETEN_RAMX2 |
     LOWPOWER_SRAMRETCTRL_RETEN_RAMX3; /* RAM instances RAM_X2 & RAM_X3 content will
     be retained */
 wakeup_interrupts = WAKEUP_RTC_LITE_ALARM_WAKEUP; /* RTC */
 wakeup_io_ctrl = 0; /* All wake-up pin disabled*/
/* Enter Deep power-down mode */
POWER_EnterDeepPowerDown (exclude_from_pd,sram_retention_ctrl, wakeup_interrupts,
     wakeup_io_ctrl);
```

14.5.2.7 Enter deep power-down mode with wake-up by OS Event Timer, using XTAL32kHz as clock source

```
/*
   * - Configure OS EVENT Timer and XTAL32kHz first, then call the sequence below
   */
   exclude_from_pd = LOWPOWER_PDCTRL0_PDEN_XTAL32K; /* The OS Event Timer will use the
      XTAL 32 kHz as clock source */
 sram_retention_ctrl = 0; /* No RAM retention */
 wakeup_interrupts = WAKEUP_OS_EVENT_TIMER; /* OS Event Timer */
 wakeup_io_ctrl = 0; /* All wake-up pin disabled*/
/* Enter deep power-down mode */
POWER_EnterDeepPowerDown (exclude_from_pd,sram_retention_ctrl, wakeup_interrupts,
     wakeup_io_ctrl);
```

14.5.2.8 Enter deep power-down mode with wake up by wake-up pin

```
/*
#define LOWPOWER_WAKEUPIOSRC_PIO0_INDEX             0  /*!< Pin P1( 1) */
#define LOWPOWER_WAKEUPIOSRC_PIO1_INDEX             2  /*!< Pin P0(28) */
#define LOWPOWER_WAKEUPIOSRC_PIO2_INDEX             4  /*!< Pin P1(18) */
#define LOWPOWER_WAKEUPIOSRC_PIO3_INDEX             6  /*!< Pin P1(30) */
#define LOWPOWER_WAKEUPIO_PIO0_PULLUPDOWN_INDEX     8  /*!< Wake-up I/O 0 pull-up/down
     configuration index */
#define LOWPOWER_WAKEUPIOSRC_DISABLE           0  /*!< Wake up is disable
     */
#define LOWPOWER_WAKEUPIOSRC_RISING            1  /*!< Wake up on rising edge
     */
#define LOWPOWER_WAKEUPIOSRC_FALLING           2  /*!< Wake up on falling edge
     */
#define LOWPOWER_WAKEUPIOSRC_RISING_FALLING         3  /*!< Wake up on both rising or
     falling edges */
#define LOWPOWER_WAKEUPIO_PIO1_PULLUPDOWN_INDEX     9  /*!< Wake-up I/O 1 pull-up/down
     configuration index */
#define LOWPOWER_WAKEUPIO_PIO2_PULLUPDOWN_INDEX    10  /*!< Wake-up I/O 2 pull-up/down
     configuration index */
#define LOWPOWER_WAKEUPIO_PIO3_PULLUPDOWN_INDEX    11  /*!< Wake-up I/O 3 pull-up/down
     configuration index */
#define LOWPOWER_WAKEUPIO_PULLDOWN    0  /*!< Select pull-down */
#define LOWPOWER_WAKEUPIO_PULLUP           1  /*!< Select pull-up     */
#define LOWPOWER_WAKEUPIO_PIO0_DISABLEPULLUPDOWN_INDEX 12  /*!< Wake-up I/O 0
     pull-up/down disable/enable control index */
#define LOWPOWER_WAKEUPIO_PIO1_DISABLEPULLUPDOWN_INDEX 13  /*!< Wake-up I/O 1
     pull-up/down disable/enable control index */
#define LOWPOWER_WAKEUPIO_PIO2_DISABLEPULLUPDOWN_INDEX 14  /*!< Wake-up I/O 2
     pull-up/down disable/enable control index */
#define LOWPOWER_WAKEUPIO_PIO3_DISABLEPULLUPDOWN_INDEX 15  /*!< Wake-up I/O 3
     pull-up/down disable/enable control index */

#define LOWPOWER_WAKEUPIO_PIO0_DISABLEPULLUPDOWN_MASK  (1UL <<
     LOWPOWER_WAKEUPIO_PIO0_DISABLEPULLUPDOWN_INDEX)  /*!< Wake-up I/O 0 pull-up/down
     disable/enable mask */
```

```
#define LOWPOWER_WAKEUPIO_PIO1_DISABLEPULLUPDOWN_MASK  (1UL <<
     LOWPOWER_WAKEUPIO_PIO1_DISABLEPULLUPDOWN_INDEX)  /*!< Wake-up I/O 1 pull-up/down
     disable/enable mask */
#define LOWPOWER_WAKEUPIO_PIO2_DISABLEPULLUPDOWN_MASK  (1UL <<
     LOWPOWER_WAKEUPIO_PIO2_DISABLEPULLUPDOWN_INDEX)  /*!< Wake-up I/O 2 pull-up/down
     disable/enable mask */
#define LOWPOWER_WAKEUPIO_PIO3_DISABLEPULLUPDOWN_MASK  (1UL <<
     LOWPOWER_WAKEUPIO_PIO3_DISABLEPULLUPDOWN_INDEX)  /*!< Wake-up I/O 3 pull-up/down
     disable/enable mask */*/

exclude_from_pd = 0; /* All modules shut down */
sram_retention_ctrl = 0; /* No RAM retained */
wakeup_interrupts = 0; /* No interrupt */

wakeup_io_ctrl = (LOWPOWER_WAKEUPIOSRC_RISING << LOWPOWER_WAKEUPIOSRC_PIO0_INDEX) |
 (LOWPOWER_WAKEUPIOSRC_FALLING << LOWPOWER_WAKEUPIOSRC_PIO1_INDEX) |
 (LOWPOWER_WAKEUPIOSRC_DISABLE << LOWPOWER_WAKEUPIOSRC_PIO2_INDEX) |
 (LOWPOWER_WAKEUPIOSRC_RISING_FALLING << LOWPOWER_WAKEUPIOSRC_PIO3_INDEX) |
     LOWPOWER_WAKEUPIO_PIO2_DISABLEPULLUPDOWN_MASK; /* with both pull-up and
     pull-down disabled. */
 /* Rising edge on wake-up pin 0, falling edge on wake-up pin 1, wake-up pin 2 disable,
 Rising edge and falling edge on wake-up pin 3 */

/* Enter Deep power-down mode */
POWER_EnterDeepPowerDown (exclude_from_pd, sram_retention_ctrl, wakeup_interrupts,
     wakeup_io_ctrl);
```

15.1 How to read this chapter

The IOCON block is included on all LPC55S6x/LPC55S2x/LPC552x devices. Registers for pins that are not available on a specific package are reserved.

Remark: Some functions, such as SCTimer/PWM inputs, frequency measure, JTAG functions, and ADC triggers are not selected through IOCON. The connections for these function are described in Chapter 18 “LPC55S6x/LPC55S2x/LPC552x Input Multiplexing (INPUTMUX)”or the chapter for the specific function. See the specific device data sheets for pinout details.

15.2 Features

All pins are standard (MFIO) port pins except P0_13 and P0_14 pins which are combo I^2C/MFIO port pins. The following electrical properties are configurable for standard port pins:

- Pull-up/pull-down resistor.
- High-speed mode.
- Open-drain mode.
- Inverted function.

Pins PIO0_13, PIO0_14, can be set either as standard port pins or as true open-drain pins that can be configured for different I^2C-bus speeds. Configuration options are somewhat different for these pins, as described in this chapter. Refer to a specific device data sheets for electrical details for these and other pins.

15.3 Basic configuration

Enable the clock to the IOCON in the AHBCLKCTRL0 register, see Table 55. Once the pins are configured, the IOCON clock can be disabled in order to conserve power.

15.4 General description

15.4.1 Pin configuration

Figure 49 shows the control of a standard GPIO pin. Even if analog switch and analog input are represented, these features are only present for some GPIO pins. When this is not the case, ASW register field exists but writing in it has no effect on the pin.

Figure 50 shows the control of a combo I²C/MFIO pin. ASW input signal is not represented since there is no analog input associated with this kind of pins for the LPC55xx. ASW register field exists but writing in it has no effect on the pin.

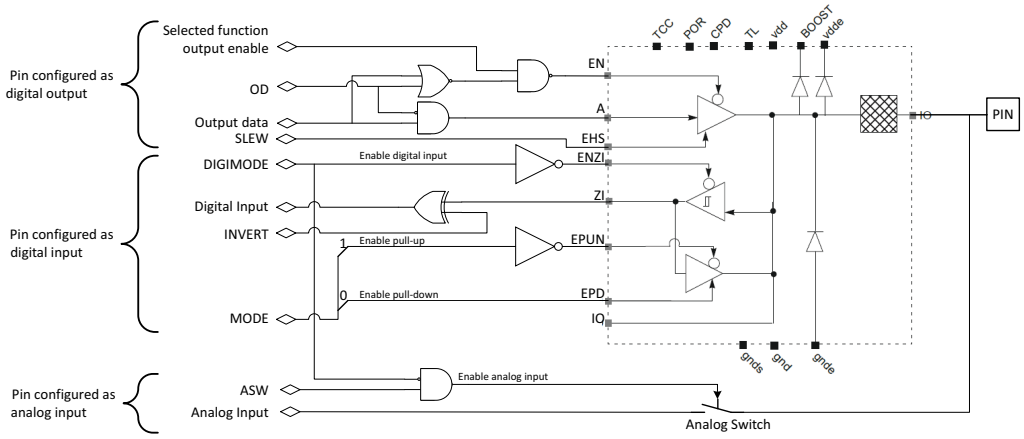


Fig 49. Standard GPIO pin configuration

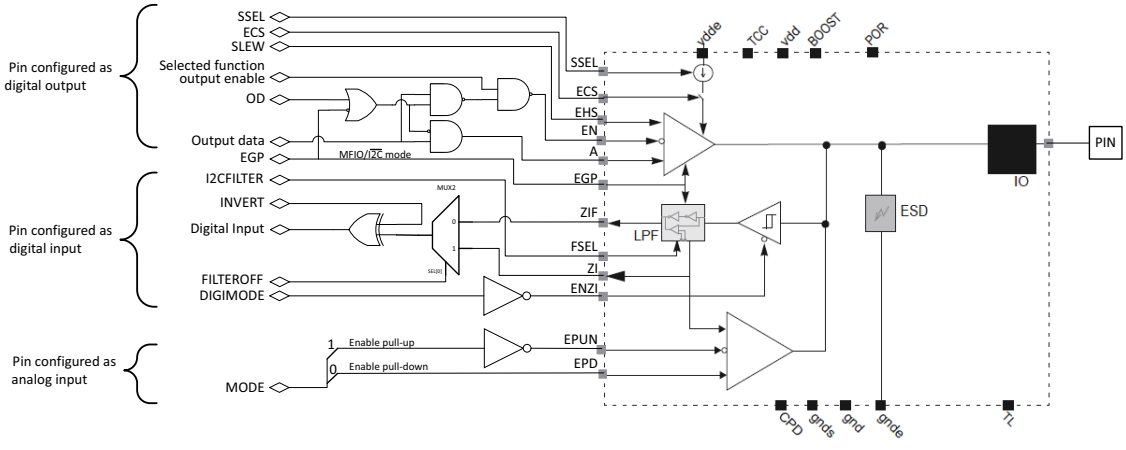


Fig 50. Combo I²C/GPIO pin configuration

15.4.2 IOCON registers

The IOCON registers control the functions and properties of device pins. Each GPIO pin has a dedicated control register to select its function and characteristics. Each pin has a unique set of functional capabilities. Not all pin characteristics are selectable on all pins. For instance, pins that have an I^2C function can be configured for different I^2C-bus modes, while pins that have an analog alternate function have an analog mode that can be selected.Details of the IOCON registers are in Section 15.5 “Register description”. The following sections describe specific characteristics of pins.

Multiple connections

Since a particular peripheral function may be allowed on more than one pin, it is possible to configure more than one pin to perform the same function. If a peripheral output function is configured to appear on more than one pin, it will in fact be routed to those pins. If a peripheral input function is defined as coming from more than one source, the values will be logically combined, possibly resulting in incorrect peripheral operation. Therefore, care should be taken to avoid this situation.

15.4.2.1 Pin function

The FUNC bits in the IOCON registers can be set to GPIO (value 0) or to a special function. The default value is FUNC = 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO block determine whether the pin is configured as an input or output see Section 16.5.3 “GPIO port direction registers”. For any special function, the pin direction is controlled automatically depending on the function. The DIR registers have no effect for special functions.

15.4.2.2 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 plain input mode or 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 for most of the pins are no pull-up/down and input disabled (tristated). Exceptions are:

- Pull-up enable, Pull-down disable, Input enable: SWDIO - P0(12), ISPSelect - P0(5).
- Pull-up disable, Pull-down enable, Input enable: SWCLK - P0(11), TRSTN - P0(2).

The repeater mode enables the pull-up resistor if the pin is high and enables the pull-down resistor if the pin is low. This causes the pin to retain its last known state if it is configured as an input and is not driven externally. Such 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.

15.4.2.3 Hysteresis

The input buffer for digital functions has built-in hysteresis. See the appropriate specific device data sheet for quantitative details.

15.4.2.4 Invert pin

This option is included to avoid having to include an external inverter on an input that is meant to be the opposite polarity of the external signal. By default this option is disabled.

15.4.2.5 Analog/digital mode

When not in digital mode (DIGIMODE = 0), a pin can be set in analog mode by setting on analog switch (ASW=1), digital input from pin is disabled and set to 0 and analog pin input is enabled. In digital mode (DIGIMODE = 1), any analog pin functions are disabled, whatever the value of ASW and digital pin functions are enabled. This protects the analog input from voltages outside the range of the analog power supply and reference that may sometimes be present on digital pins. All pin types include this control, even if they do not support any analog functions. However, the digital output is not automatically disabled, so the pin output enable must be deactivated by selecting an input function (FUNC field).

In order to use a pin that has an analog input (ADC or Comparator) option for that purpose, select GPIO function (FUNC field = 0), set this GPIO in input mode (DIRPi[j] = 0 or DIRCLRPi[j]=1; see Section 16.5.3 “GPIO port direction registers”, disable the digital pin function (DIGIMODE = 0) and enable the analog switch (ASW=1). The MODE field should also be set to 0.

In analog mode, the MODE field should be “Inactive” (00); the INVERT, FILTEROFF, and OD settings have no effect. For an unconnected pin that has an analog function, keep the ASW bit set to 0 (analog input disabled), disable the digital input (DIGIMODE=0) and select plain input mode (no pull-up nor pull-down mode) in the MODE field. It isolates the pin from the circuit inside and saves power.

15.4.2.6 Input filter

Some pins include a filter that can be selectively disabled by setting the FILTEROFF bit. This concerns combo I^2C/MFIO pins. The filter suppresses input pulses smaller than about 3 ns in MFIO mode and smaller than 10 ns or 50 ns in I^2C mod, depending on the value of I^2CFILTER field.

15.4.2.7 Output slew rate

The SLEW bits of digital outputs that do not need to switch state should be set to “standard”. This setting allows multiple outputs to switch simultaneously without noticeably degrading the power/ground distribution of the device, and has a small effect on signal transition time. This is particularly important if analog accuracy is significant to the application. See the relevant specific device data sheet for more details.

15.4.2.8 I^2C modes

Pins that support I^2C with specialized pad electronics (PIO0_13 and PIO0_14) have additional configuration bits. These have multiple configurations to support I^2C variants. These are not hard-wired so that the pins can be easily used for non-I^2C functions. See Table 336 for recommended mode settings.

For non-I^2C operation, these pins can be open-drain or not, as standard (MFIO) pins.

15.4.2.9 Open-drain mode

When output is selected, either by selecting a special function in the FUNC field, or by selecting the GPIO function for a pin having a 1 in the related bit of that port's DIR register, a 1 in the OD bit selects open-drain operation, that is, a 1 disables the high-drive transistor. This option has no effect on the combo I^2C/MFIO pins when I^2C mode but has same effect as standard pin when in MFIO mode. Note that the properties of a pin in this simulated open-drain mode are somewhat different than those of a true open drain output.

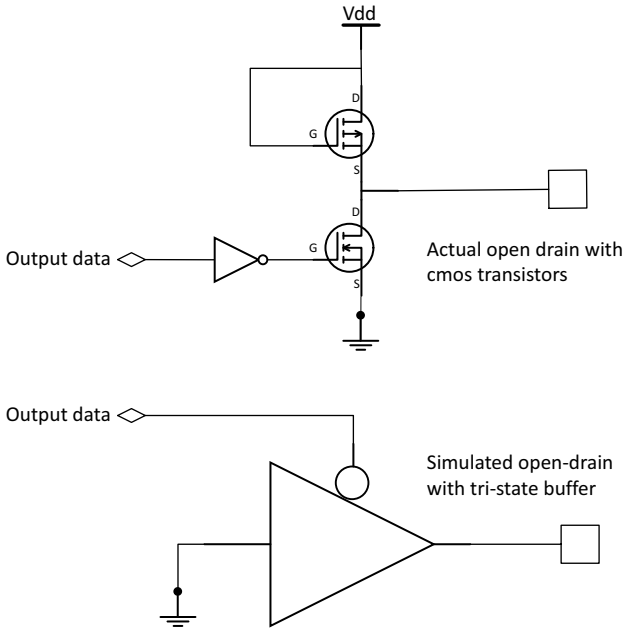


Fig 51. Open drain mode

15.5 Register description

Each port pin PIOm_n has one IOCON register assigned to control the electrical characteristics of the pin.

Remark: See the pinning information section of the appropriate device data sheet for details on which pins listed in Table 333 exist on each package configuration.

One may identify three pin types:

- Digital only pin (D).
- Analog/digital pins (A).
- I^2C pin (I).

Table 333 gives an overview of IOCON registers. All of them are 32-bit RW registers. Some bit fields are not used and are reserved.

Table 333. Register overview: I/O configuration (base address = 0x4000 1000)

Offset	Register	Access	Pin type	Reset value	Section
0x000	Digital I/O control for port 0 pins PIO0_0 (PIO0_0).	RW	A	0x0000	15.5.3
0x004	Digital I/O control for port 0 pins PIO0_1 (PIO0_1).	RW	D	0x0000	15.5.1
0x008	Digital I/O control for port 0 pins PIO0_2 (PIO0_2).	RW	D	0x0110	15.5.1
0x00C - 0x010	Digital I/O control for port 0 pins PIO0_b (PIO0_3 - PIO0_4).	RW	D	0x0000	15.5.1
0x014	Digital I/O control for port 0 pins PIO0_5 (PIO0_5).	RW	D	0x0120	15.5.1
0x018 - 0x020	Digital I/O control for port 0 pins PIO0_b (PIO0_6 - PIO0_8).	RW	D	0x0000	15.5.1
0x024 - 0x028	Digital I/O control for port 0 pins PIO0_b (PIO0_9 - PIO0_10).	RW	A	0x0000	15.5.3
0x02C	Digital I/O control for port 0 pins PIO0_11 (PIO0_11).	RW	A	0x0116	15.5.3
0x030	Digital I/O control for port 0 pins PIO0_12 (PIO0_12).	RW	A	0x0126	15.5.3
0x034 - 0x038	Digital I/O control for port 0 pins PIO0_b (PIO0_13 - PIO0_14).	RW	I	0x5000	15.5.2
0x03C - 0x040	Digital I/O control for port 0 pins PIO0_b (PIO0_15 - PIO0_16).	RW	A	0x0000	15.5.3
0x044	Digital I/O control for port 0 pins PIO0_17 (PIO0_17).	RW	D	0x0000	15.5.1
0x048	Digital I/O control for port 0 pins PIO0_18 (PIO0_18).	RW	A	0x0000	15.5.3
0x04C - 0x058	Digital I/O control for port 0 pins PIO0_b (PIO0_19 - PIO0_22).	RW	D	0x0000	15.5.1
0x05C	Digital I/O control for port 0 pins PIO0_23 (PIO0_23).	RW	A	0x0000	15.5.3
0x060 - 0x078	Digital I/O control for port 0 pins PIO0_b (PIO0_24 - PIO0_30).	RW	D	0x0000	15.5.1
0x07C - 0x080	Digital I/O control for port a pins PIOa_b (PIO0_31 - PIO1_0).	RW	A	0x0000	15.5.3
0x084 - 0x09C	Digital I/O control for port 1 pins PIO1_b (PIO1_1 - PIO1_7).	RW	D	0x0000	15.5.1
0x0A0 - 0x0A4	Digital I/O control for port 1 pins PIO1_b (PIO1_8 - PIO1_9).	RW	A	0x0000	15.5.3
0x0A8 - 0x0B4	Digital I/O control for port 1 pins PIO1_b (PIO1_10 - PIO1_13).	RW	D	0x0000	15.5.1
0x0B8	Digital I/O control for port 1 pins PIO1_14 (PIO1_14).	RW	A	0x0000	15.5.3
0x0BC - 0x0C8	Digital I/O control for port 1 pins PIO1_b (PIO1_15 - PIO1_18).	RW	D	0x0000	15.5.1
0x0CC	Digital I/O control for port 1 pins PIO1_19 (PIO1_19).	RW	A	0x0000	15.5.3
0x0D0 - 0x0FC	Digital I/O control for port 1 pins PIO1_b (PIO1_20 - PIO1_31).	RW	D	0x0000	15.5.1

15.5.1 Type D IOCON registers

Table 334 applies to pins referenced as pin type D in Table 333.

Reset values concern all pin of this type except PIO0_2 and PIO0_5 (see notes [1] and [2]).

Table 334. Type D IOCON registers

Bit	Symbol	Value	Description	Reset value
3:0	FUNC	-	Selects pin function. See Table 340, Table 341, and Table 342.	0
5:4	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0 [1]
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
6	SLEW		Driver slew rate.	0
		0	Standard-mode, output slew rate is slower. More outputs can be switched simultaneously.	
		1	Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.	
7	INVERT		Input polarity.	0
		0	Disabled. Input function is not inverted.	
		1	Enabled. Input function is inverted.	
8	DIGIMODE		Digital mode enable or disable	0[2]
		0	Disable digital mode. Digital input set to 0.	
		1	Enable digital mode. Digital input enabled.	
9	OD		Controls open-drain mode.	0
		0	Normal. Normal push-pull output	
		1	Open-drain. Simulated open-drain output (high drive disabled).	
31:10	-	-	Reserved. Read value is undefined, only zero should be written.	NA

[1] Except for pin PIO0_2 where MODE = 1 (pull-down) and PIO0_5 where MODE = 2 (pull-up).

[2] Except PIO0_2 and PIO0_5 where DIGIMODE = 1 (Digital input enabled).

15.5.2 Type I IOCON registers

Table 335 applies to pins PIO0_13 and PIO0_14. See Table 336 for recommended setting for I^2C operation.

Table 335. Type I IOCON registers

Bit	Symbol	Value	Description	Reset value
3:0	FUNC	-	Selects pin function. See Table 340 “I/O control registers: FUNC values (FUNC = 0 to 4) and pin functions”, Table 341 “I/O control registers: FUNC values (FUNC = 5 to 9) and pin functions”, and Table 342 “I/O control registers: FUNC values (FUNC = 10 to 15) and pin functions”.	0

Table 335. Type I IOCON registers *…continued*

Bit	Symbol	Value	Description	Reset value
5:4	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
6	SLEW		Driver slew rate.	0
		0	Standard-mode, output slew rate is slower. More outputs can be switched simultaneously.	
		1	Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.	
7	INVERT	1	Input polarity.	0
		0	Disabled. Input function is not inverted.	
		1	Enabled. Input is function inverted.	
8	DIGIMODE		Digital mode enable or disable	0
		0	Disable digital mode. Digital input is set to 0.	
		1	Enable digital mode. Digital input is enabled.	
9	OD		Controls open-drain mode in standard GPIO mode (EGP = 1). This bit has no effect in I^2C mode (EGP=0).	0
		0	Normal. Normal push-pull output.	
		1	Open-drain. Simulated open-drain output (high drive disabled).	
10	-	-	Reserved. Read value is undefined, only zero should be written.	NA
11	SSEL		Supply selection bit.	0
		0	3V3 signaling in I^2C mode.	
		1	1V8 signaling in I^2C mode.	
12	FILTEROFF		Controls input glitch filter.	1
		0	Filter enabled. Noise pulses below approximately 3 ns are filtered out in GPIO mode (EGP = 1). In I^2C mode (EGP = 0), noise pulses below approximately 10 ns or 50 ns are filtered out, depending on I^2CFILTER bit field value.	
		1	Filter disabled. No input filtering is done.	
13	ECS		Pull-up current source enable in I^2C mode.	0
		0	Disabled. IO is an open drain cell.	
		1	Enabled. Pull-up resistor is connected.	
14	EGP		Switch between GPIO mode and I^2C mode.	1
		0	I^2C mode.	
		1	GPIO mode.	
15	I2CFILTER		Configures I^2C features for Standard-mode, Fast-mode, Fast-mode Plus operation and High-Speed mode operation	0
		0	I^2C 50 ns glitch filter enabled. Typically used for Standard mode, Fast-mode and Fast-mode Plus I^2C.	
		1	I^2C 10 ns glitch filter enabled. Typically used for High-Speed mode I^2C.	
31:16	-	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 336. Suggested IOCON settings for I²C functions

Mode	IOCON register bit						
	15: I²CFILTER	14: I²CDRIVE	13: ECS	12: FILTEROFF	8: DIGIMODE	7: INVERT	6: SLEW
GPIO low-speed mode	-	1	-	0 [1]	1 [2]	0	0
GPIO high-speed mode	-	1	-	1 [1]	1 [2]	0	1
Standard-mode I²C	1	0	0	0	1	0	0
Fast-mode Plus I²C	0	0	0	0	1	0	0
High-speed slave I²C	-	0	1	1	1	0	0

[1] The input filter may be turned off by setting FILTEROFF if it is not needed.

[2] The input may be turned off by clearing DIGIMODE if it is not needed.

15.5.3 Type A IOCON registers

Table 337 applies to pins referenced as pin type A in Table 333.

Reset values all pins of this type except PIO0_11 and PIO0_12 (see notes [1] and [2]).

Table 337. Type A IOCON registers

Bit	Symbol	Value	Description	Reset value
3:0	FUNC	-	Selects pin function. See Table 340, Table 341, and Table 342.	0[2]
5:4	MODE	-	Selects function mode (on-chip pull-up/pull-down resistor control).	0 [1]
		0x0	Inactive input (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
6	SLEW		Driver slew rate.	0
		0	Standard-mode, output slew rate is slower. More outputs can be switched simultaneously.	
		1	Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.	
7	INVERT		Input polarity.	0
		0	Disabled. Input function is not inverted.	
		1	Enabled. Input is function inverted.	
8	DIGIMODE		Digital mode enable or disable.	0
		0	Disable digital mode. Digital input set to 0.	
		1	Enable digital mode. Digital input enabled.	
9	OD		Controls open-drain mode.	0
		0	Normal. Normal push-pull output.	
		1	Open-drain. Simulated open-drain output (high drive disabled).	

Table 337. Type A IOCON registers *...continued*

Bit	Symbol	Value	Description	Reset value
10	ASW		Analog switch input control. Usable only if DIGIMODE = 0.	0
		0	**For device revision 0A:** For pins PIO0_9, PIO0_11, PIO0_12, PIO0_15, PIO0_18, PIO0_31, PIO1_0 and PIO1_9, analog switch is closed (enabled). For the other pins, analog switch is open (disabled). **For device revision 1B:** Analog switch is open (disabled).	
		1	**For device revision 0A:** For all pins except PIO0_9, PIO0_11, PIO0_12, PIO0_15, PIO0_18, PIO0_31, PIO1_0 and PIO1_9 analog switch is closed (enabled) **For device revision 1B:** Analog switch is closed (enabled).	
31:11	-	-	Reserved. Read value is undefined, only zero should be written.	NA

[1] Except PIO0_11 and PIO0_12 where FUNC = 6 (rep. functions SWCLK and SDIO).

[2] Except for pin PIO0_11 where MODE = 1 (pull-down) and PIO0_12 where MODE = 2 (pull-up).

[3] Except PIO0_11 and PIO0_12 where DIGIMODE = 1 (Digital input enabled).

To enable an analog input, select the GPIO function, disable the digital functions of the pin by clearing the DIGIMODE bit in the related IOCON register and:

For device revision 0A:

- For pins PIO0_9, PIO0_11, PIO0_12, PIO0_15, PIO0_18, PIO0_31, PIO1_0 and P31, PIO1_0 and PIO1_9, leave ASW bit at '0' in the related IOCON register.
- For the other pins, set the ASW bit to '1' in the related IOCON register.

For device revision 1B:

- Set the ASW bit to '1' in the related IOCON register.

In GPIO module, set the related GPIO bit direction (DIR) as input. Table 338 shows the analog input related to each pin of type A and Table 339 gives the pin associated to each analog input function.

Table 338. Analog functions sorted by pin numbers

Pin	Analog function	Pin	Analog function
PIO0_0	ACMP0_A	P1_14	ACMP0_D
PIO0_9	ACMP0_B	P1_19	$ACMPV_{REF}$
PIO0_10	ADC0_1	P2_0	ADC0_5
PIO0_11	ADC0_9	P2_1	ADC0_13
PIO0_12	ADC0_10	P2_11	ADC0_7
PIO0_15	ADC0_2	P2_12	ADC0_15
PIO0_16	ADC0_8	P2_13	ADC0_6
PIO0_18	ACMP0_C	P2_14	ADC0_14
PIO0_23	ADC0_0	P2_23	ACMP0_E

Table 338. Analog functions sorted by pin numbers *...continued*

Pin	Analog function	Pin	Analog function
PIO0_31	ADC0_3	-	-
PIO1_0	ADC0_11	-	-
PIO1_8	ADC0_4	-	-
PIO1_9	ADC0_12	-	-

Table 339. Analog inputs sorted by function types

ADC input	Pin	Comparator input	Pin
ADC0_0	P0_23	ACMP0_A	P0_0
ADC0_1	P0_10	ACMP0_B	P0_9
ADC0_2	P0_15	ACMP0_C	P0_18
ADC0_3	P0_31	ACMP0_D	P1_14
ADC0_4	P1_8	ACMP0_E	P2_23
ADC0_5	P2_0	$ACMPV_{REF}$	P1_19
ADC0_6	P2_13		
ADC0_7	P2_11		
ADC0_8	P0_16		
ADC0_9	P0_11		
ADC0_10	P0_12		
ADC0_11	P1_0		
ADC0_12	P1_9		
ADC0_13	P2_1		
ADC0_14	P2_14		
ADC0_15	P2_12		

The FUNC field for PIO0_11 and PIO0_12 resets to 0b110 (0x6), selecting the Serial Wire Debug function by default (SWCLK and SDIO).

15.5.4 IOCON pin functions in relation to FUNC values

Table 340, Table 341, and Table 342 show the functions associated to each pin. FUNC value controls the function that is connected to the pin.

Table 340. I/O control registers: FUNC values (FUNC = 0 to 4) and pin functions

Reg name/ FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4
P0_0	-	FC3_SCK	CT0MAT0	SCT_GPI0
P0_1	-	FC3_CTS_SDAX_SSEL0	CT_INP0	SCT_GPI1
P0_2	FC3_TXD_SCL_MISO	CT_INP1	SCT0_OUT0	SCT_GPI2
P0_3	FC3_RXD_SDA_MOSI	CT0MAT1	SCT0_OUT1	SCT_GPI3
P0_4	-	FC4_SCK	CT_INP12	SCT_GPI4
P0_5	-	FC4_RXD_SDA_MOSI	CT3MAT0	SCT_GPI5
P0_6	FC3_SCK	CT_INP13	CT4MAT0	SCT_GPI6
P0_7	FC3_RTS_SCLX_SSEL1	SD0_CLK	FC5_SCK	FC1_SCK
P0_8	FC3_SSEL3	SD0_CMD	FC5_RXD_SDA_MOSI	SWO

Table 340. I/O control registers: FUNC values (FUNC = 0 to 4) and pin functions *…continued*

Reg name/ FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4
P0_9	FC3_SSEL2	SD0_POW_EN	FC5_TXD_SCL_MISO	-
P0_10	FC6_SCK	CT_INP10	CT2MAT0	FC1_TXD_SCL_ MISO
P0_11	FC6_RXD_SDA_MOSI	CT2MAT2	FREQME_GPIO_CLK_A	-
P0_12	FC3_TXD_SCL_MISO	SD1_BACKEND_PWR	FREQME_GPIO_CLK_B	SCT_GPI7
P0_13	FC1_CTS_SDAX_SSEL0	UTICK_CAP0	CT_INP0	SCT_GPI0
P0_14	FC1_RTS_SCLX_SSEL1	UTICK_CAP1	CT_INP1	SCT_GPI1
P0_15	FC6_CTS_SDAX_SSEL0	UTICK_CAP2	CT_INP16	SCT0_OUT2
P0_16	FC4_TXD_SCL_MISO	CLKOUT	CT_INP4	-
P0_17	FC4_SSEL2	SD0_CARD_DET_N	SCT_GPI7	SCT0_OUT0
P0_18	FC4_CTS_SDAX_SSEL0	SD0_WR_PRT	CT1MAT0	SCT0_OUT1
P0_19	FC4_RTS_SCLX_SSEL1	UTICK_CAP0	CT0MAT2	SCT0_OUT2
P0_20	FC3_CTS_SDAX_SSEL0	CT1MAT1	CT_INP15	SCT_GPI2
P0_21	FC3_RTS_SCLX_SSEL1	UTICK_CAP3	CT3MAT3	SCT_GPI3
P0_22	FC6_TXD_SCL_MISO	UTICK_CAP1	CT_INP15	SCT0_OUT3
P0_23	MCLK	CT1MAT2	CT3MAT3	SCT0_OUT4
P0_24	FC0_RXD_SDA_MOSI	SD0_D(0)	CT_INP8	SCT_GPI0
P0_25	FC0_TXD_SCL_MISO	SD0_D(1)	CT_INP9	SCT_GPI1
P0_26	FC2_RXD_SDA_MOSI	CLKOUT	CT_INP14	SCT0_OUT5
P0_27	FC2_TXD_SCL_MISO	-	CT3MAT2	SCT0_OUT6
P0_28	FC0_SCK	SD1_CMD	CT_INP11	SCT0_OUT7
P0_29	FC0_RXD_SDA_MOSI	SD1_D(2)	CT2MAT3	SCT0_OUT8
P0_30	FC0_TXD_SCL_MISO	SD1_D(3)	CT0MAT0	SCT0_OUT9
P0_31	FC0_CTS_SDAX_SSEL0	SD0_D(2)	CT0MAT1	SCT0_OUT3
P1_0	FC0_RTS_SCLX_SSEL1	SD0_D(3)	CT_INP2	SCT_GPI4
P1_1	FC3_RXD_SDA_MOSI	-	CT_INP3	SCT_GPI5
P1_2	-	-	CT0MAT3	SCT_GPI6
P1_3	-	-	-	SCT0_OUT4
P1_4	FC0_SCK	SD0_D(0)	CT2MAT1	SCT0_OUT0
P1_5	FC0_RXD_SDA_MOSI	SD0_D(2)	CT2MAT0	SCT_GPI0
P1_6	FC0_TXD_SCL_MISO	SD0_D(3)	CT2MAT1	SCT_GPI3
P1_7	FC0_RTS_SCLX_SSEL1	SD0_D(1)	CT2MAT2	SCT_GPI4
P1_8	FC0_CTS_SDAX_SSEL0	SD0_CLK	-	SCT0_OUT1
P1_9	-	FC1_SCK	CT_INP4	SCT0_OUT2
P1_10	-	FC1_RXD_SDA_MOSI	CT1MAT0	SCT0_OUT3
P1_11	-	FC1_TXD_SCL_MISO	CT_INP5	USB0_VBUS
P1_12	-	FC6_SCK	CT1MAT1	USB0_PORTPWRN
P1_13	-	FC6_RXD_SDA_MOSI	CT_INP6	USB0_ OVERCURRENTN
P1_14	-	UTICK_CAP2	CT1MAT2	FC5_CTS_SDAX_ SSEL0

Table 340. I/O control registers: FUNC values (FUNC = 0 to 4) and pin functions *…continued*

Reg name/ FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4
P1_15	-	UTICK_CAP3	CT_INP7	FC5_RTS_SCLX_ SSEL1
P1_16	-	FC6_TXD_SCL_MISO	CT1MAT3	SD0_CMD
P1_17	-	-	FC6_RTS_SCLX_SSEL1	SCT0_OUT4
P1_18	SD1_POW_EN	-	-	SCT0_OUT5
P1_19	-	SCT0_OUT7	CT3MAT1	SCT_GPI7
P1_20	FC7_RTS_SCLX_SSEL1	-	CT_INP14	-
P1_21	FC7_CTS_SDAX_SSEL0	-	CT3MAT2	-
P1_22	-	SD0_CMD	CT2MAT3	SCT_GPI5
P1_23	FC2_SCK	SCT0_OUT0	SD1_D(3)	-
P1_24	FC2_RXD_SDA_MOSI	SCT0_OUT1	SD1_D(1)	-
P1_25	FC2_TXD_SCL_MISO	SCT0_OUT2	SD1_D(0)	UTICK_CAP0
P1_26	FC2_CTS_SDAX_SSEL0	SCT0_OUT3	CT_INP3	UTICK_CAP1
P1_27	FC2_RTS_SCLX_SSEL1	SD0_D(4)	CT0MAT3	CLKOUT
P1_28	FC7_SCK	SD0_D(5)	CT_INP2	-
P1_29	FC7_RXD_SDA_MOSI	SD0_D(6)	SCT_GPI6	USB1_PORTPWRN
P1_30	FC7_TXD_SCL_MISO	SD0_D(7)	SCT_GPI7	USB1_ OVERCURRENTN
P1_31	MCLK	SD1_CLK	CT0MAT2	SCT0_OUT6

Table 341. I/O control registers: FUNC values (FUNC = 5 to 9) and pin functions

Pin	FUNC = 5	FUNC = 6	FUNC = 7	FUNC = 8	FUNC = 9
P0_0	-	SD1_CARD_INT_N	-	-	-
P0_1	-	SD1_CLK	CMP0_OUT	-	-
P0_2	-	-	-	-	-
P0_3	-	-	-	-	-
P0_4	-	-	-	FC3_CTS_SDAX _ SSEL0	-
P0_5	-	-	-	FC3_RTS_SCLX_ SSEL1	MCLK
P0_6	-	-	-	-	-
P0_7	-	-	-	-	-
P0_8	-	-		-	-
P0_9	-	-	-	-	-
P0_10	SCT0_OUT2	SWO	-	-	-
P0_11	-	SWCLK	-	-	-
P0_12	SD0_POW_EN	SWDIO	FC6_TXD_SCL_ MISO	-	-
P0_13	FC1_RXD_SDA_MOSI	-	-	-	PLU_IN0
P0_14	-	FC1_TXD_SCL_MISO	-	-	PLU_IN1
P0_15	SD0_WR_PRT	-	-	-	-

Table 341. I/O control registers: FUNC values (FUNC = 5 to 9) and pin functions *…continued*

Pin	FUNC = 5	FUNC = 6	FUNC = 7	FUNC = 8	FUNC = 9
P0_16	-	-	-	-	-
P0_17	-	-	-	SD0_CARD_INT_N	PLU_IN2
P0_18	-	-	-	-	PLU_IN3
P0_19		-	FC7_TXD_SCL_MISO	-	PLU_IN4
P0_20	-	-	FC7_RXD_SDA_MOSI	HS_SPI_SSEL0	PLU_IN5
P0_21	-	-	FC7_SCK	-	PLU_CLKIN
P0_22	-	-	USB0_VBUS	SD1_D(0)	PLU_OUT7
P0_23	FC0_CTS_SDAX_SSEL0	-	-	SD1_D(1)	-
P0_24	-	-	-	-	-
P0_25	-	-	-	-	-
P0_26	-	-	USB0_IDVALUE	FC0_SCK	HS_SPI_MOSI
P0_27	-	-	FC7_RXD_SDA_MOSI	-	PLU_OUT0
P0_28	-	-	USB0_OVERCURRENTN	-	PLU_OUT1
P0_29	-	-	CMP0_OUT	-	PLU_OUT2
P0_30	-	-		-	-
P0_31	-	-	-	-	-
P1_0	-	-	-	-	PLU_OUT3
P1_1	HS_SPI_SSEL1	-	USB1_OVERCURRENTN	-	PLU_OUT4
P1_2	-	HS_SPI_SCK	USB1_PORTPWRN	-	PLU_OUT5
P1_3	-	HS_SPI_MISO	USB0_PORTPWRN	-	PLU_OUT6
P1_4	FREQME_GPIO_CLK_A	-	-	-	-
P1_5	-	-	-	-	-
P1_6	-	-	-	-	-
P1_7	-	-	-	-	-
P1_8	FC4_SSEL2	-	-	-	-
P1_9	FC4_CTS_SDAX_SSEL0	-	-	-	-
P1_10	-	-	-	-	-
P1_11	-	-	-	-	-
P1_12	HS_SPI_SSEL2	-	-	-	-
P1_13	USB0_FRAME	-	SD0_CARD_DET_N	-	-
P1_14	USB0_LEDN	-	SD1_CMD	-	-
P1_15	FC4_RTS_SCLX_SSEL1	-	SD1_D(2)	-	-
P1_16	-	-	-	-	-
P1_17	-	-	SD1_CARD_INT_N	-	SD1_CARD_DET_N

Table 341. I/O control registers: FUNC values (FUNC = 5 to 9) and pin functions *…continued*

Pin	FUNC = 5	FUNC = 6	FUNC = 7	FUNC = 8	FUNC = 9
P1_18	-	-	PLU_OUT0	-	-
P1_19	FC4_SCK	-	PLU_OUT1	-	-
P1_20	FC4_TXD_SCL_MISO	-	PLU_OUT2	-	-
P1_21	FC4_RXD_SDA_MOSI		PLU_OUT3	-	-
P1_22	FC4_SSEL3	-	PLU_OUT4	-	-
P1_23	FC3_SSEL2	-	PLU_OUT5	-	-
P1_24	FC3_SSEL3	-	PLU_OUT6	-	-
P1_25	-	-	PLU_CLKIN	-	-
P1_26	HS_SPI_SSEL3	-	PLU_IN5	-	-
P1_27	-	-	PLU_IN4	-	-
P1_28	-	-	PLU_IN3	-	-
P1_29	USB1_FRAME	-	PLU_IN2	-	-
P1_30	USB1_LEDN	-	PLU_IN1	-	-
P1_31	-	-	PLU_IN0	-	-

Table 342. I/O control registers: FUNC values (FUNC = 10 to 15) and pin functions

Pin	FUNC = 10	FUNC = 11
P0_0	P0_SEC(0)	-
P0_1	P0_SEC(1)	-
P0_2	P0_SEC(2)	-
P0_3	P0_SEC(3)	-
P0_4	P0_SEC(4)	-
P0_5	P0_SEC(5)	-
P0_6	P0_SEC(6)	-
P0_7	P0_SEC(7)	-
P0_8	P0_SEC(8)	-
P0_9	P0_SEC(9)	-
P0_10	P0_SEC(10)	-
P0_11	P0_SEC(11)	-
P0_12	P0_SEC(12)	-
P0_13	P0_SEC(13)	-
P0_14	P0_SEC(14)	-
P0_15	P0_SEC(15)	-
P0_16	P0_SEC(16)	-
P0_17	P0_SEC(17)	-
P0_18	P0_SEC(18)	-
P0_19	P0_SEC(19)	-
P0_20	P0_SEC(20)	FC4_TXD_SCL_MISO
P0_21	P0_SEC(21)	-
P0_22	P0_SEC(22)	-
P0_23	P0_SEC(23)	-

Table 342. I/O control registers: FUNC values (FUNC = 10 to 15) and pin functions *…continued*

Pin	FUNC = 10	FUNC = 11
P0_24	P0_SEC(24)	-
P0_25	P0_SEC(25)	-
P0_26	P0_SEC(26)	-
P0_27	P0_SEC(27)	-
P0_28	P0_SEC(28)	-
P0_29	P0_SEC(29)	-
P0_30	P0_SEC(30)	-
P0_31	P0_SEC(31)	-

16.1 How to read this chapter

GPIO registers support up to 32 pins on each port. Depending on the device and package type, a subset of those pins may be available, and the unused bits in GPIO registers are reserved. See Table 343.

Table 343. GPIO pins available

Package	Total GPIOs	GPIO Port 0	GPIO Port 1
LQFP100 and VFBGA98	64	PIO0_0 to PIO0_31	PIO1_0 to PIO1_31
LQFP 64	36	PIO0_0 to PIO0_31	PIO1_0 to PIO1_3

16.2 Features

- GPIO pins can be configured as input or output by software.
- Most GPIO pins default to tri-state but there are a few that are set to pull-up or pull-down as shown in Section 15.4.2.2 “Pin mode”. Interrupts are disabled at reset.
- Pin registers allow pins to be sensed and set individually.
- Direction (input/output) can be set and cleared individually. Pins can be masked in input for security purposes. See: Section 47.3.4 “Interrupt, DMA and GPIO: Secure instance and masking”

16.3 Basic configuration

For the GPIO port registers, enable the clock to each GPIO port in the AHBCLKCTRL0 register Table 55.

16.4 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 Chapter 19 “LPC55S6x/LPC55S2x/LPC552x Pin Interrupt and Pattern Match (PINT)” and Chapter 21 “LPC55S6x/LPC55S2x/LPC552x:Group GPIO Input Interrupt (GINT0/1)”.

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.

16.5 Register description

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

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

Remark: A reset value noted as “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 344. Register overview: GPIO port (base address 0x4008 C000)

Name	Access	Offset	Description	Reset value	Section
B0_[31:0]	R/W	[0x0000:0x001F]	Byte pin registers for ports 0 GPIO pins.	ext	16.5.1
B1_[31:0]	R/W	[0x0020:0x003F]	Byte pin registers for ports 1 GPIO pins.	ext	16.5.1
W0_[31:0]	R/W	[0x1000:0x107C]	Word pin registers for ports 0 GPIO pins.	ext	16.5.2
W1_[31:0]	R/W	[0x1080:0x10FC]	Word pin registers for ports 1 GPIO pins.	ext	16.5.2
DIR0	R/W	0x2000	Direction registers port 0	0	16.5.3
DIR1	R/W	0x2004	Direction registers port 1	0	16.5.3
MASK0	R/W	0x2080	Mask register port 0.	0	16.5.4
MASK1	R/W	0x2084	Mask register port 1.	0	16.5.4
PIN0	R/W	0x2100	Port pin register port 0	ext	16.5.5
PIN1	R/W	0x2104	Port pin register port 1	ext	16.5.5
MPIN0	R/W	0x2180	Masked port register port 0.	ext	16.5.6
MPIN1	R/W	0x2184	Masked port register port 1.	ext	16.5.6
SET0	R/W	0x2200	Write: Set register for port 0. Read: output bits for port 0.	0	16.5.7
SET1	R/W	0x2204	Write: Set register for port 1. Read: output bits for port 1.	0	16.5.7
CLR0	WO	0x2280	Clear port 0.	NA	16.5.8
CLR1	WO	0x2284	Clear port 1.	NA	16.5.8
NOT0	WO	0x2300	Toggle port 0.	NA	16.5.9
NOT1	WO	0x2304	Toggle port 1.	NA	16.5.9
DIRSET0	WO	0x2380	Set pin direction bits for port 0.	0	16.5.10
DIRSET1	WO	0x2384	Set pin direction bits for port 1.	0	16.5.10
DIRCLR0	WO	0x2400	Clear pin direction bits for port 0.	-	16.5.11
DIRCLR1	WO	0x2404	Clear pin direction bits for port 1.	-	16.5.11
DIRNOT0	WO	0x2480	Toggle pin direction bits for port 0.	-	16.5.12
DIRNOT1	WO	0x2484	Toggle pin direction bits for port 1.	-	16.5.12

16.5.1 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 half words to sense or set the state of two pins, and read or write words to sense or set the state of four pins.

Table 345. GPIO port byte pin registers (Ba_b, a = 0 to 1, b = 0 to 31, offset 0h + (a × 20h) + (b × 1h))

Bit	Symbol	Description	Reset value
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. Write: loads the output bit of the pin. **Remark:** One register for each port pin. Supported pins depends on the specific device and package.	ext
7:1		Reserved (0 on read, ignored on write)	undefined

16.5.2 GPIO port word pin registers

Each GPIO pin has a word register in this address range. Any byte, half word, 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 346. GPIO port word pin registers (Wa_b, a = 0 to 1, b = 0 to 31, offsets 1000h + (a × 80h) + (b × 4h))

Bit	Symbol	Description	Reset value
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. Only 0 or 0xFFFF FFFF can be read. Writing any value other than 0 will set the output bit. **Remark:** One register for each port pin. Supported pins depend on the specific device and package.	ext

16.5.3 GPIO port direction registers

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

Table 347. GPIO direction port register (DIRa, a = 0…1, offset 2000h + (a × 4h))

Bit	Symbol	Description	Reset value
31:0	DIRP	Selects pin direction for pin PIOm_n (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). 0 = input. 1 = output. **Remark:** Supported pins depends on the specific device and package.	0x0

16.5.4 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 348. GPIO mask port register (MASKa, a = 0…1, offset 2080h + (a × 4h))

Bit	Symbol	Description	Reset value
31:0	MASKP	Controls which bits corresponding to PIOm_n are active in the MPORT register (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). 0 = Read MPORT: pin state, write MPORT: load output bit. 1 = Read MPORT: 0, write MPORT: output bit not affected. **Remark:** Supported pins depends on the specific device and package.	0x0

16.5.5 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 349. GPIO port pin register (PINa , a = 0…1, offset 2100h + (a × 4h))

Bit	Symbol	Description	Reset value
31:0	PORT	Reads pin states or loads output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). 0 = Read: pin is low, write: clear output bit. 1 = Read: pin is high, write: set output bit. **Remark:** Supported pins depends on the specific device and package.	ext

16.5.6 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 350. GPIO masked port pin register (MPINa , a = 0…1, offset 2180h + (a × 4h))

Bit	Symbol	Description	Reset value
31:0	MPORTP	Masked port register (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). 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. **Remark:** Supported pins depends on the specific device and package.	ext

16.5.7 GPIO port set register

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 351. GPIO set port register (SETa, a = 0…1, offset 2200h + (a × 4h))

Bit	Symbol	Description	Reset value
31:0	SETP	Read or set output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). 0 = Read: output bit: write: no operation. 1 = Read: output bit, write: set output bit. **Remark:** Supported pins depends on the specific device and package.	0x0

16.5.8 GPIO port clear register

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

Table 352. GPIO clear port register (CLRa, a = 0…1, offset 2280h + (a × 4h))

Bit	Symbol	Description	Reset value
31:0	CLRP	Clear output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). 0 = No operation. 1 = Clear output bit. **Remark:** Supported pins depends on the specific device and package.	N/A

16.5.9 GPIO port toggle register

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

Table 353. GPIO toggle port register (NOTa, a = 0…1, offset 2300h + (a × 4h))

Bit	Symbol	Description	Reset value
31:0	NOTP	Toggle output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). 0 = no operation. 1 = Toggle output bit. **Remark:** Supported pins depends on the specific device and package.	N/A

16.5.10 GPIO port direction set register

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

Table 354. GPIO port direction set register (DIRSETa, a = 0….1 offset 2380h + (a × 4h))

Bit	Symbol	Description	Reset value
31:0	DIRSETP	Set direction bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). 0 = No operation. 1 = Set direction bit. **Remark:** Supported pins depends on the specific device and package.	N/A

16.5.11 GPIO port direction clear register

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

Table 355. GPIO port direction clear register (DIRCLRa, a = 0....1, offset 2400h = (a x 4h))

Bit	Symbol	Description	Reset value
31:0	DIRCLRP	Clear direction bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). 0 = No operation. 1 = Clear direction bit. **Remark:** Supported pins depends on the specific device and package.	undefined

16.5.12 GPIO port direction toggle register

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

Table 356. GPIO port direction toggle register (DIRNOTa, a = 0....1, offset 2480h + (a x 4h))

Bit	Symbol	Description	Reset value
31:0	DIRNOTP	Toggle direction bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). 0 = no operation. 1 = Toggle direction bit. **Remark:** Supported pins depends on the specific device and package.	undefined

16.6 Functional description

16.6.1 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 need to be selected for GPIO in *I/O Configuration* to read its state. However, the *Input Enable* bit of the pad must be set in *I/O Configuration* otherwise its value is 0. There are four ways to read the pin state:

- The state of a single pin can be read with seven high-order zeros from a Byte Pin register.
- The state of a single pin can be read in all bits of a byte, half word, 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.
- Each pin input can be masked independently of each other for security purpose. When a pin is masked, its read state is 0. See Section 47.3.4 "Interrupt, DMA and GPIO: Secure instance and masking".

16.6.2 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. To set the output of the GPIO pin, use the following steps.

1. The pin must be selected for GPIO operation via IOCON (this is the default except for 2 pins: P0_11 and P0_12).
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 the output bits of a port can be read from its SET register. Reading any of the registers described in returns the state of pins, regardless of their direction or alternate functions. See Section 16.6.1 "Reading pin state".

16.6.3 Masked I/O

The MASK register of a port 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.

16.6.4 GPIO direction

Each pin in a GPIO port can be configured as input or output using the DIR registers. The direction of individual pins can be set, cleared, or toggled using the DIRSET, DIRCLR, and DIRNOT registers.

16.6.5 Recommended practices

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

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

17.1 How to read this chapter

Secure GPIO registers support up to the 32 pins on port 0. Each of the 32 Secure GPIO are associated to a specific function (P0_SEC(i), i = 0…31) in IOCON module.

17.2 Features

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

17.3 Basic configuration

For the Secure GPIO registers, enable the clock to Secure GPIO in the SYSCON AHBCLKCTRL2 register, see Table 57.

17.4 General description

The Secure 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 Secure GPIOs can be used as external interrupts together with the secure pin block, see Chapter 20 “LPC55S6x/LPC55S2x/LPC552x Secure pin interrupt and pattern match (Secure PINT)” and Chapter 18 “LPC55S6x/LPC55S2x/LPC552x Input Multiplexing (INPUTMUX)”.

The Secure GPIO registers configure each Secure 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. When configured in output, for the bit value to be driven to the pin, FUNC = 10 must be set in IOCON PIO0_n register, see Chapter 15 “LPC55S6x/LPC55S2x/LPC552x I/O Pin Configuration (IOCON)”.

17.5 Register description

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

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

Remark: A reset value noted as *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 357. Register overview: Secure GPIO port (base address 0x400A 8000)

Name	Access	Offset	Description	Reset value	Section
B[31:0]	R/W	[0x0000:0x001F]	Byte pin registers.	ext	17.5.1
W[31:0]	R/W	[0x1000:0x12D8]	Word pin registers.	ext	17.5.2
DIR	R/W	0x2000	Direction register.	0	17.5.3
MASK	R/W	0x2080	Mask register.	0	17.5.4
PIN	R/W	0x2100	Port pin register.	ext	17.5.5
MPIN	R/W	0x2180	Masked port register.	ext	17.5.6
SET	R/W	0x2200	Write: Set register. Read: output bits.	0	17.5.7
CLR	WO	0x2280	Clear register.	NA	17.5.8
NOT	WO	0x2300	Toggle register.	NA	17.5.9
DIRSET	WO	0x2380	Set pin direction bits.	0	17.5.10
DIRCLR	WO	0x2400	Clear pin direction bits.	NA	17.5.11
DIRNOT	WO	0x2480	Toggle pin direction bits.	NA	17.5.12

17.5.1 Secure GPIO port byte pin registers

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

Table 358. GPIO port byte pin registers (B0_n, n=0 to 31, offset 0h + (n × 1h))

Bit	Symbol	Description	Reset value	Access
0	PBYTE	Read: state of the pin PIO0_n, regardless of direction, masking, or alternate function, except that pins configured as analog I/O always read as 0 (DIGIMODE = 0 in IOCON P0_n register). Write: loads the pin's output bit. For the bit value to be driven to the pin, FUNC = 10 must be set in IOCON PIO0_n register. **Remark:** One register for each port0 pin.	ext	R/W
7:1	-	Reserved (0 on read, ignored on write)	0	-

17.5.2 Secure GPIO port word pin registers

Each Secure 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 359. Secure GPIO port word pin registers (W0_n, n=0 to 31, offsets 1000h + (n × 4h))

Bit	Symbol	Description	Reset value	Access
31:0	PWORD	Read 0: pin PIO_SEC(n) is LOW. Write 0: clear output bit (FUNC = 10 must be set in IOCON P0_n register) Read 0xFFFF FFFF: pin PIO_SEC(n) is HIGH. Write any value 0x0000 0001 to 0xFFFF FFFF: set output bit (FUNC = 10 must be set in IOCON P0_n register). **Remark:** Only 0 or 0xFFFF FFFF can be read. Writing any value other than 0 will set the output bit. One register for each port0 pin.	ext	R/W

17.5.3 Secure GPIO port direction register

Direction register for configuring the pins as inputs or outputs.

Table 360. Secure GPIO direction port register (DIR, offset 2000h)

Bit	Symbol	Description	Reset value	Access
31:0	DIRP	Selects pin direction for pin PIO_SEC(n) (bit 0 = PIO_SEC(0), bit 1 = PIO_SEC(1), etc.). 0 = input. 1 = output.	0	R/W

17.5.4 Secure GPIO port mask register

This register affects writing and reading the MPORT register. Zeroes in this register enable reading and writing; ones disable writing and result in zeros in corresponding positions when reading.

Table 361. Secure GPIO mask port register (MASK, offset 2080h)

Bit	Symbol	Description	Reset value	Access
31:0	MASKP	Controls which bits corresponding to PIO_SEC(n) are active in the MPORT register (bit 0 = PIO_SEC(0), bit 1 = PIO_SEC(1), etc.,). 0 = Read MPORT: pin state; write MPORT: load output bit. 1 = Read MPORT: 0; write MPORT: output bit not affected.	0	R/W

17.5.5 Secure GPIO port pin register

Reading this register 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 (i.e., when DIGIMODE = 0 in IOCON P0_n register). Writing this register loads the output bits of the pins written to, regardless of the Mask register. FUNC = 10 must be set in IOCON P0_n corresponding registers for the bits value to be driven to the pins.

Table 362. Secure GPIO port pin register (PIN, offset 2100h)

Bit	Symbol	Description	Reset value	Access
31:0	PORT	Reads pin states or loads output bits (bit 0 = PIO_SEC(0), bit 1 = PIO_SEC(1), etc.,). 0 = Read: pin is low; write: clear output bit. 1 = Read: pin is high; write: set output bit.	ext	R/W

17.5.6 Secure GPIO masked port pin register

This register is similar to the PORT register, except that the value read is masked by ANDing with the inverted contents of the corresponding MASK register, and writing to one of this register only affects output register bits that are enabled by zeros in the MASK register. FUNC = 10 must be set in the IOCON P0_n corresponding registers for the bits value to be transmitted to the pins.

Table 363. Secure GPIO masked port pin register (MPIN, offset 2180h)

Bit	Symbol	Description	Reset value	Access
31:0	MPORTP	Masked port register (bit 0 = PIO_SEC(0), bit 1 = PIO_SEC(1), etc.,). 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

17.5.7 Secure GPIO port set register

Output bits can be set by writing ones to this register, regardless of MASK register. FUNC = 10 must be set in IOCON P0_n corresponding registers for the bits value to be driven to the pins. Reading from this register returns the port's output bits, regardless of pin directions.

Table 364. Secure GPIO set port register (SET, offset 2200h)

Bit	Symbol	Description	Reset value	Access
31:0	SETP	Read or set output bits (bit 0 = PIO_SEC(0), bit 1 = PIO_SEC(1), etc.,). 0 = Read: output bit: write: no operation. 1 = Read: output bit; write: set output bit.	0	R/W

17.5.8 Secure GPIO port clear register

Output bits can be cleared by writing ones to this write-only register, regardless of MASK register. FUNC = 10 must be set in IOCON P0_n corresponding registers for the bits value to be driven to the pins.

Table 365. Secure GPIO clear port register (CLR, offset 2280h)

Bit	Symbol	Description	Reset value	Access
31:0	CLRP	Clear output bits (bit 0 = PIO_SEC(0), bit 1 = PIO_SEC(1), etc.,). 0 = No operation. 1 = Clear output bit.	NA	WO

17.5.9 Secure GPIO port toggle register

Output bits can be toggled/inverted/complemented by writing ones to this write-only register, regardless of MASK register. FUNC = 10 must be set in IOCON P0_n corresponding registers for the bits value to be driven to the pins.

Table 366. Secure GPIO toggle port register (NOT, offset 2300h)

Bit	Symbol	Description	Reset value	Access
31:0	NOTP	Toggle output bits (bit 0 = PIO_SEC(0), bit 1 = PIO_SEC(1), etc.,). 0 = no operation. 1 = Toggle output bit.	NA	WO

17.5.10 Secure GPIO port direction set register

Direction bits can be set by writing ones to this register.

Table 367. Secure GPIO port direction set register (DIRSET, offset 2380h)

Bit	Symbol	Description	Reset value	Access
31:0	DIRSETP	Set direction bits (bit 0 = PIO_SEC(0), bit 1 = PIO_SEC(1), etc.,). 0 = No operation. 1 = Set direction bit.	0	WO

17.5.11 Secure GPIO port direction clear register

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

Table 368. Secure GPIO port direction clear register (DIRCLR, offset 2400h)

Bit	Symbol	Description	Reset value	Access
31:0	DIRCLRP	Clear direction bits (bit 0 = PIO_SEC(0), bit 1 = PIO_SEC(1), etc.). 0 = No operation. 1 = Clear direction bit.	NA	WO

17.5.12 Secure GPIO port direction toggle register

Direction bits can be set by writing ones to this write-only register.

Table 369. Secure GPIO port direction toggle register (DIRNOT, offset 2480h)

Bit	Symbol	Description	Reset value	Access
31:0	DIRNOTP	Toggle direction bits (bit 0 = PIO_SEC(0), bit 1 = PIO_SEC(1), etc.). 0 = no operation. 1 = Toggle direction bit.	NA	WO

17.6 Functional description

17.6.1 Reading pin state

Software can read the state of all Secure GPIO pins except those selected for analog input or output in the *I/O Configuration* logic. A pin does not need to be selected for Secure GPIO in *I/O Configuration* in order to read its state. However, the *Input Enable* bit (DIGIMODE) of the pad must be set in *I/O Configuration* otherwise the pin state value would be 0. There are four ways to read pin state:

- The state of a single pin can be read with seven 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 can be read as a byte, halfword, or word from the PORT register.
- The state of a selected subset of the pin can be read from the Masked Port (MPORT) register. Pins having a 1 in the port's Mask register will read as 0 from the MPORT register

17.6.2 Secure GPIO output

Each secure GPIO pin has an output bit in the secure 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 secure GPIO operation via IOCON (FUNC value must be 10).
2. The pin must be selected for output by a 1 in the 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 secure 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 the PORT register loads the output bits of all the pins written to.
- Writing to the MPORT register loads the output bits of pins identified by zeros in corresponding positions of the MASK register.
- Writing ones to the SET register sets output bits.
- Writing ones to the CLR register clears output bits.
- Writing ones to the NOT register toggles/complements/inverts output bits.

The state of the output bits can be read from the SET register. Reading any of the registers described in Section 17.6.1 returns the state of pins, regardless of their direction or alternate functions.

17.6.3 Masked I/O

The MASK register defines which of its pins should be accessible in the 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 the MASK register contains all zeros, the PORT and MPORT registers operate identically for reading and writing.

Applications in which interrupts can result in masked secure GPIO operation, or in task switching among tasks that do masked secure 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 the 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 register, and set/capture the semaphore controlling exclusive use of the MASK register before setting the MASK register, and release the semaphore after the last operation that uses the MPORT or MASK registers.

17.6.4 Secure GPIO direction

Each pin can be configured as input or output using the DIR registers. The direction of individual pins can be set, cleared, or toggled using the DIRSET, DIRCLR, and DIRNOT registers.

17.6.5 Recommended practices

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

- For initial setup after reset or re-initialization, write the PORT register.
- 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 the PORT register.

18.1 How to read this chapter

Input multiplexing is present on all LPC55S6x/LPC55S2x/LPC552x devices. Depending on the package, not all inputs from external pins may be available.

18.2 Features

- Configures the inputs to the SCT.
- Configures the inputs to the asynchronous CTimers.
- Configures the inputs to the pin interrupt block and pattern match engine.
- Configures the inputs to the pin interrupt secure block and pattern match engine.
- Configures the inputs to the DMA0 and DMA1 triggers.
- Enables the inputs to the DMA0 and DMA1 requests.
- Configures the inputs to the frequency measure function. This function is controlled by the FREQMECTRL register in Chapter 11 “LPC55S6x/LPC55S2x/LPC552x Analog control”.

18.3 Basic configuration

Once set up, no clocks are required for the input multiplexer to function. The system clock is needed only to write to or read from the INPUT MUX registers. Once the input multiplexer is configured, disable the clock to the INPUT MUX block in the AHBCLKCTRL register. See: Section 4.5.17 “AHB clock control 0”.

18.4 Pin description

The input multiplexer has no dedicated pins. However, all digital pins of ports 0 and 1 can be selected as inputs to the pin interrupts. Multiplexer inputs from external pins work independently of any other function assigned to the pin as long as no analog function is enabled.

Table 370. INPUT MUX pin description

Pins	Peripheral	Section
Any existing pin on port 0 or 1	Pin interrupts 0 to 7	18.6.3
PIO0_11, PIO0_12, PIO1_4, PIO2_7	Frequency measure block	18.6.9
SCT0_GPI [0:7] pin functions selected from IOCON register (See the Pin descriptions in LPC55xx data sheet).	SCTimer/PWM	Chapter 24 “LPC55S6x/LPC55S2x/LPC552x SCTimer/PWM (SCT)”

18.5 General description

Some peripheral inputs are multiplexed to multiple input sources. The sources can be external pins, interrupts, output signals of other peripherals, or other internal signals.

Input multiplexers for most peripherals consist of one or more multiplexers that choose one of several inputs to be routed to a specific input of that peripheral. For example, each CTimer has four capture inputs, each of which has an input multiplexer that can select from among a number of pin functions (if they are connected via IOCON) and some internal signals.

Figure 52 shows a generic input multiplexer arrangement with n inputs. For example, in the case of the CTimers, there will be four of these for each timer, one for each capture input.

Some peripherals have a more complex arrangement and have detailed figures. See Section 18.5.1 “SCT0 input multiplexing” for SCT/PWM input multiplexing, Section 18.5.2 “Pin interrupt input multiplexing” for Pin interrupt (PINT) multiplexing, and Section 18.5.4 “DMA trigger input multiplexing” for DMA trigger multiplexing.

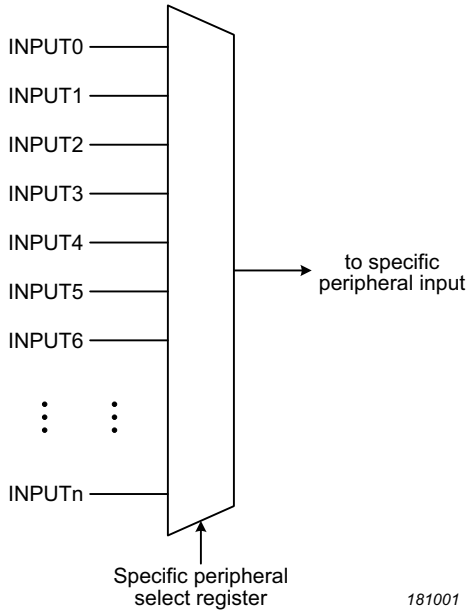


Fig 52. Generic input multiplexing

18.5.1 SCT0 input multiplexing

The input multiplexing for the SCT0 timer multiplexes between 24 internal or external sources for each of its 7 outputs. These outputs with the pll_clk are the 8 inputs of the SCT_TIMER. Figure 53 shows the detail of this multiplexing.

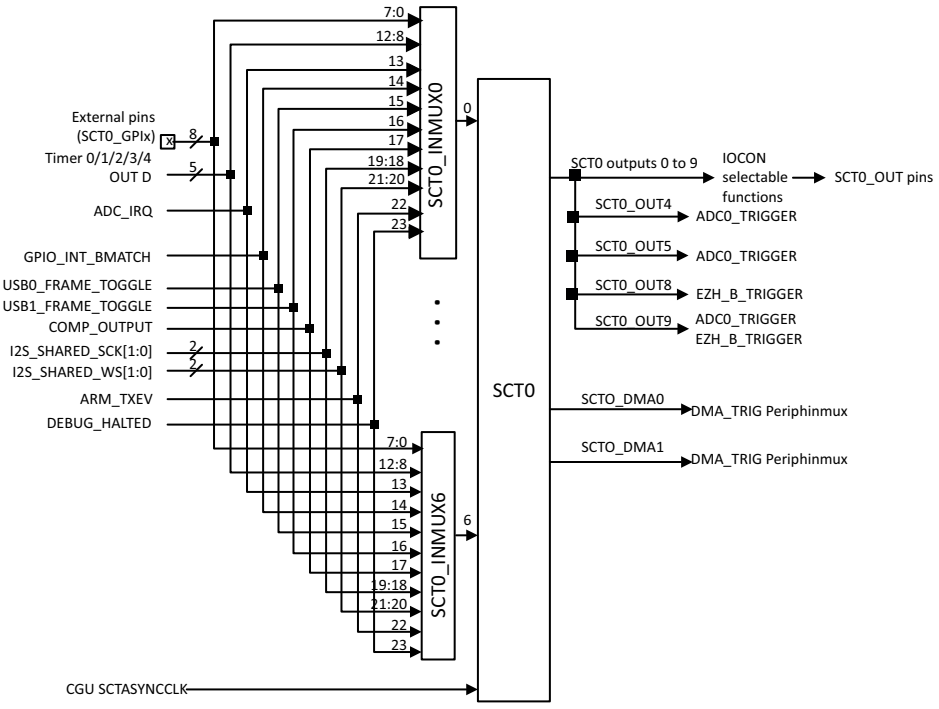


Fig 53. SCT0 input multiplexing

18.5.2 Pin interrupt input multiplexing

The input multiplexing for the pin interrupts and pattern match engine multiplexes all existing pins from ports 0 and 1.

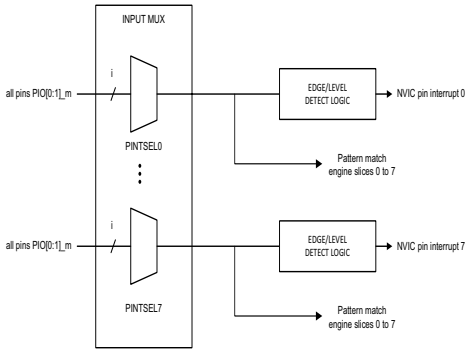


Fig 54. Pin interrupt multiplexing

18.5.3 Pin interrupt secure input multiplexing

The input multiplexing for the pin interrupts secure and pattern match engine multiplexes all existing pins from ports 0 (with function 10, P0_SEC, selected)

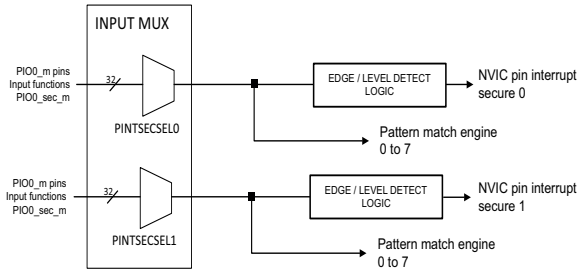


Fig 55. Pin interrupt secure multiplexing

18.5.4 DMA trigger input multiplexing

The trigger input multiplexing for each DMA controller is configured as shown in Figure 56. In each DMA controller, four of these input triggers are selected from the DMA trigger outputs, controlled by the DMA_OTRIG_INMUX registers. See Section 18.6.6 "DMA0 output trigger feedback multiplexing registers 0 to 3" and Section 18.6.8 "DMA1 output trigger feedback multiplexing registers 0 to 3".

The potential trigger selections for DMA0 are shown in Section 18.6.5 "DMA0 trigger input multiplexing registers 0 to 22 ". The potential trigger selections for DMA1 are shown in Section 18.6.7 "DMA1 trigger input multiplexing registers 0 to 9".

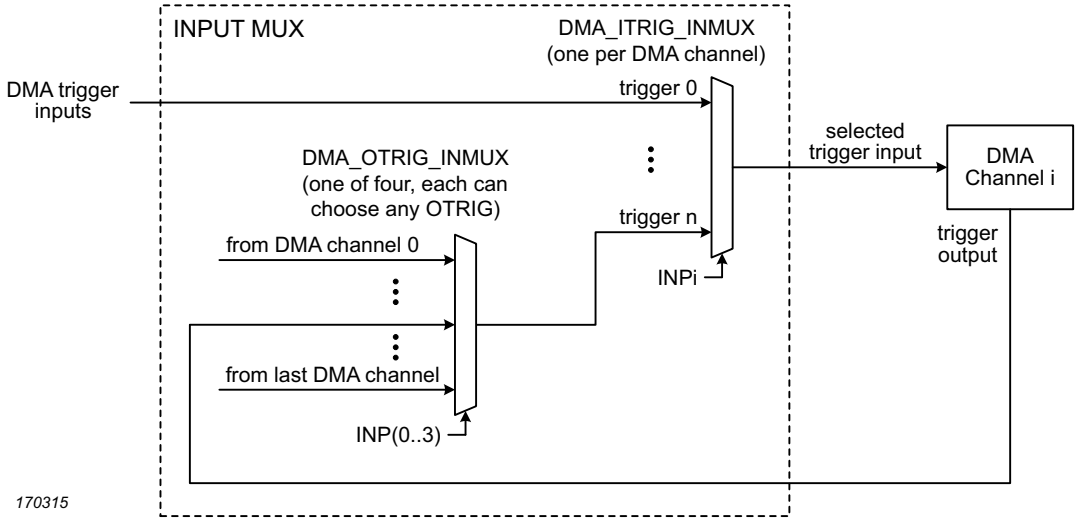


Fig 56. DMA trigger input multiplexing

The two DMA controllers, DMA0 and DMA1, each receive the same DMA requests from peripherals on the first 10 requests. DMA request enables are provided to allow controlling which DMA controller (if any) receives each request.

18.6 Register description

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

All address offsets not shown or empty in Table 371 are reserved and should not be written.

Table 371. Register overview: INPUTMUX (base address = 0x50006000)

Name	Access	Offset	Description	Reset value	Section
SCT0_INMUX0	R/W	0x00	Input mux register for SCT0 input 0.	0x1F	18.6.1
SCT0_INMUX1	R/W	0x004	Input mux register for SCT0 input 1.	0x1F	18.6.1
SCT0_INMUX2	R/W	0x008	Input mux register for SCT0 input 2.	0x1F	18.6.1
SCT0_INMUX3	R/W	0x00C	Input mux register for SCT0 input 3.	0x1F	18.6.1
SCT0_INMUX4	R/W	0x010	Input mux register for SCT0 input 4.	0x1F	18.6.1
SCT0_INMUX5	R/W	0x014	Input mux register for SCT0 input 5.	0x1F	18.6.1
SCT0_INMUX6	R/W	0x018	Input mux register for SCT0 input 6.	0x1F	18.6.1
TIMER0CAPTSEL0	R/W	0x020	Capture select registers for TIMER0 inputs 0.	0x1F	18.6.2
TIMER0CAPTSEL1	R/W	0x024	Capture select registers for TIMER0 inputs 1.	0x1F	18.6.2
TIMER0CAPTSEL2	R/W	0x028	Capture select registers for TIMER0 inputs 2.	0x1F	18.6.2
TIMER0CAPTSEL3	R/W	0x02C	Capture select registers for TIMER0 inputs 3.	0x1F	18.6.2
TIMER1CAPTSEL0	R/W	0x040	Capture select registers for TIMER1 inputs 0.	0x1F	18.6.2
TIMER1CAPTSEL1	R/W	0x044	Capture select registers for TIMER1 inputs 1.	0x1F	18.6.2
TIMER1CAPTSEL2	R/W	0x048	Capture select registers for TIMER1 inputs 2.	0x1F	18.6.2
TIMER1CAPTSEL3	R/W	0x04C	Capture select registers for TIMER1 inputs 3.	0x1F	18.6.2
TIMER2CAPTSEL0	R/W	0x060	Capture select registers for TIMER2 inputs 0.	0x1F	18.6.2
TIMER2CAPTSEL1	R/W	0x064	Capture select registers for TIMER2 inputs 1.	0x1F	18.6.2
TIMER2CAPTSEL2	R/W	0x068	Capture select registers for TIMER2 inputs 2.	0x1F	18.6.2
TIMER2CAPTSEL3	R/W	0x06C	Capture select registers for TIMER2 inputs 3.	0x1F	18.6.2
PINTSEL0	R/W	0x0C0	Pin interrupt select register 0.	0x7F	18.6.3
PINTSEL1	R/W	0x0C4	Pin interrupt select register 1.	0x7F	18.6.3
PINTSEL2	R/W	0x0C8	Pin interrupt select register 2.	0x7F	18.6.3
PINTSEL3	R/W	0x0CC	Pin interrupt select register 3.	0x7F	18.6.3
PINTSEL4	R/W	0x0D0	Pin interrupt select register 4.	0x7F	18.6.3
PINTSEL5	R/W	0x0D4	Pin interrupt select register 5.	0x7F	18.6.3
PINTSEL6	R/W	0x0D8	Pin interrupt select register 6.	0x7F	18.6.3
PINTSEL7	R/W	0x0DC	Pin interrupt select register 7.	0x1F	18.6.3
DMA0_ITRIG_INMUX0	R/W	0x0E0	Trigger select register for DMA0 channel 0.	0x1F	18.6.5
DMA0_ITRIG_INMUX1	R/W	0x0E4	Trigger select register for DMA0 channel 1.	0x1F	18.6.5
DMA0_ITRIG_INMUX2	R/W	0x0E8	Trigger select register for DMA0 channel 2.	0x1F‘	18.6.5
DMA0_ITRIG_INMUX3	R/W	0x0EC	Trigger select register for DMA0 channel 3.	0x1F	18.6.5
DMA0_ITRIG_INMUX4	R/W	0x0F0	Trigger select register for DMA0 channel 4.	0x1F	18.6.5
DMA0_ITRIG_INMUX5	R/W	0x0F4	Trigger select register for DMA0 channel 5.	0x1F	18.6.5
DMA0_ITRIG_INMUX6	R/W	0x0F8	Trigger select register for DMA0 channel 6.	0x1F	18.6.5

Table 371. Register overview: INPUTMUX (base address = 0x50006000) *…continued*

Name	Access	Offset	Description	Reset value	Section
DMA0_ITRIG_INMUX7	R/W	0x0FC	Trigger select register for DMA0 channel 7.	0x1F	18.6.5
DMA0_ITRIG_INMUX8	R/W	0x100	Trigger select register for DMA0 channel 8.	0x1F	18.6.5
DMA0_ITRIG_INMUX9	R/W	0x104	Trigger select register for DMA0 channel 9.	0x1F	18.6.5
DMA0_ITRIG_INMUX10	R/W	0x108	Trigger select register for DMA0 channel 10.	0x1F	18.6.5
DMA0_ITRIG_INMUX11	R/W	0x10C	Trigger select register for DMA0 channel 11.	0x1F	18.6.5
DMA0_ITRIG_INMUX12	R/W	0x110	Trigger select register for DMA0 channel 12.	0x1F	18.6.5
DMA0_ITRIG_INMUX13	R/W	0x114	Trigger select register for DMA0 channel 13.	0x1F	18.6.5
DMA0_ITRIG_INMUX14	R/W	0x118	Trigger select register for DMA0 channel 14.	0x1F	18.6.5
DMA0_ITRIG_INMUX15	R/W	0x11C	Trigger select register for DMA0 channel 15.	0x1F	18.6.5
DMA0_ITRIG_INMUX16	R/W	0x120	Trigger select register for DMA0 channel 16.	0x1F	18.6.5
DMA0_ITRIG_INMUX17	R/W	0x124	Trigger select register for DMA0 channel 17.	0x1F	18.6.5
DMA0_ITRIG_INMUX18	R/W	0x128	Trigger select register for DMA0 channel 18.	0x1F	18.6.5
DMA0_ITRIG_INMUX19	R/W	0x12C	Trigger select register for DMA0 channel 19.	0x1F	18.6.5
DMA0_ITRIG_INMUX20	R/W	0x130	Trigger select register for DMA0 channel 20.	0x1F	18.6.5
DMA0_ITRIG_INMUX21	R/W	0x134	Trigger select register for DMA0 channel 21.	0x1F	18.6.5
DMA0_ITRIG_INMUX22	R/W	0x138	Trigger select register for DMA0 channel 22.	0x1F	18.6.5
DMA0_OTRIG_INMUX0	R/W	0x160	DMA0 output trigger selection for DMA0 trigger 0.	0x1F	18.6.6
DMA0_OTRIG_INMUX1	R/W	0x164	DMA0 output trigger selection for DMA0 trigger 1.	0x1F	18.6.6
DMA0_OTRIG_INMUX2	R/W	0x168	DMA0 output trigger selection for DMA0 trigger 2.	0x1F	18.6.6
DMA0_OTRIG_INMUX3	R/W	0x16C	DMA0 output trigger selection for DMA0 trigger 3.	0x1F	18.6.6
FREQMEAS_REF	R/W	0x180	Frequency measurement reference clock select	0xF	18.6.9
FREQMEAS_TARGET	R/W	0x184	Frequency measurement target clock select	0xF	18.6.10
TIMER3CAPTSEL0	R/W	0x1A0	Capture select registers for TIMER3 inputs	0x1F	18.6.2
TIMER3CAPTSEL1	R/W	0x1A4	Capture select registers for TIMER3 inputs	0x1F	18.6.2
TIMER3CAPTSEL2	R/W	0x1A8	Capture select registers for TIMER3 inputs	0x1F	18.6.2
TIMER3CAPTSEL3	R/W	0x1AC	Capture select registers for TIMER3 inputs	0x1F	18.6.2
TIMER4CAPTSEL0	R/W	0x1C0	Capture select registers for TIMER4 inputs	0x1F	18.6.2
TIMER4CAPTSEL1	R/W	0x1C4	Capture select registers for TIMER4 inputs	0x1F	18.6.2
TIMER4CAPTSEL2	R/W	0x1C8	Capture select registers for TIMER4 inputs	0x1F	18.6.2
TIMER4CAPTSEL3	R/W	0x1CC	Capture select registers for TIMER4 inputs	0x1F	18.6.2
PINTSECSEL0	R/W	0x1E0	Pin interrupt secure select	0x3F	18.6.4
PINTSECSEL1	R/W	0x1E4	Pin interrupt secure select	0x3F	18.6.4
DMA1_ITRIG_INMUX0	R/W	0x200	Trigger select register for DMA1 channel 0	0xF	18.6.7
DMA1_ITRIG_INMUX1	R/W	0x204	Trigger select register for DMA1 channel 1	0xF	18.6.7
DMA1_ITRIG_INMUX2	R/W	0x208	Trigger select register for DMA1 channel 2	0xF	18.6.7
DMA1_ITRIG_INMUX3	R/W	0x20C	Trigger select register for DMA1 channel 3	0xF	18.6.7
DMA1_ITRIG_INMUX4	R/W	0x210	Trigger select register for DMA1 channel 4	0xF	18.6.7
DMA1_ITRIG_INMUX5	R/W	0x214	Trigger select register for DMA1 channel 5	0xF	18.6.7
DMA1_ITRIG_INMUX6	R/W	0x218	Trigger select register for DMA1 channel 6	0xF	18.6.7
DMA1_ITRIG_INMUX7	R/W	0x21C	Trigger select register for DMA1 channel 7	0xF	18.6.7
DMA1_ITRIG_INMUX8	R/W	0x220	Trigger select register for DMA1 channel 8	0xF	18.6.7

Table 371. Register overview: INPUTMUX (base address = 0x50006000) *...continued*

Name	Access	Offset	Description	Reset value	Section
DMA1_ITRIG_INMUX9	R/W	0x224	Trigger select register for DMA1 channel 9	0xF	18.6.7
DMA1_OTRIG_INMUX0	R/W	0x240	DMA1 output trigger selection for DMA1 trigger	0xF	18.6.8
DMA1_OTRIG_INMUX1	R/W	0x244	DMA1 output trigger selection for DMA1 trigger	0xF	18.6.8
DMA1_OTRIG_INMUX2	R/W	0x248	DMA1 output trigger selection for DMA1 trigger	0xF	18.6.8
DMA1_OTRIG_INMUX3	R/W	0x24C	DMA1 output trigger selection for DMA1 trigger	0xF	18.6.8
DMA0_REQ_ENA	R/W	0x740	Enable DMA0 requests	0x7FFFFF	18.6.11.1
DMA0_REQ_ENA_SET	W	0x748	Set one or several bits in DMA0_REQ_ENA	0x7FFFFF	18.6.11.2
DMA0_REQ_ENA_CLR	W	0x750	Clear one or several bits in DMA0_REQ_ENA	0x7FFFFF	18.6.11.3
DMA1_REQ_ENA	R/W	0x760	Enable DMA1 requests	0x7FFFFF	18.6.11.4
DMA1_REQ_ENA_SET	W	0x768	Set one or several bits in DMA1_REQ_ENA	0x7FFFFF	18.6.11.5
DMA1_REQ_ENA_CLR	W	0x770	Clear one or several bits in DMA1_REQ_ENA	0x7FFFFF	18.6.11.6
DMA0_ITRIG_ENA	R/W	0x780	Enable DMA0 triggers	0x7FFFFF	18.6.11.7
DMA0_ITRIG_ENA_SET	W	0x788	Set one or several bits in DMA0_ITRIG_ENA	0x7FFFFF	18.6.11.8
DMA0_ITRIG_ENA_CLR	W	0x790	Clear one or several bits in DMA0_ITRIG_ENA	0x7FFFFF	18.6.11.9
DMA1_ITRIG_ENA	R/W	0x7A0	Enable DMA1 triggers	0x7FFFFF	18.6.11.10
DMA1_ITRIG_ENA_SET	W	0x7A8	Set one or several bits in DMA1_ITRIG_ENA	0x7FFFFF	18.6.11.11
DMA1_ITRIG_ENA_CLR	W	0x7B0	Clear one or several bits in DMA1_ITRIG_ENA	0x7FFFFF	18.6.11.12

18.6.1 SCT0 Input multiplexing registers 0 to 6

With the SCT0 Input multiplexing registers you can select one input source for each SCT0 input from 24 external and internal sources. (An exception is SCT0 input SCT0_IN7, which is directly connected to the SCTASYNCCLK PLL clock and not multiplexed with any other signals.)

The output of SCT0 Input multiplexing register 0 selects the source for SCT0 input 0. The output of SCT0 Input multiplexing register 1 selects the source for SCT0 input 1, and so forth up to SCT0 Input multiplexing register 6, which selects the input for SCT0 input 6.

Table 372. SCT0 Input multiplexing registers 0 to 6 (SCT0_INMUX[0:6], offset [0x000: 0x018])

Bit	Symbol	Access	Value	Description	Reset value
4:0	INP_N	RW		Input number to SCT0 inputs 0 to 6.	0x1F
			0	SCT_GPI0 function selected from IOCON register.	
			1	SCT_GPI1 function selected from IOCON register.	
			2	SCT_GPI2 function selected from IOCON register.	
			3	SCT_GPI3 function selected from IOCON register.	
			4	SCT_GPI4 function selected from IOCON register.	
			5	SCT_GPI5 function selected from IOCON register.	
			6	SCT_GPI6 function selected from IOCON register.	
			7	SCT_GPI7 function selected from IOCON register.	
			8	T0_OUT0 ctimer 0 match[0] output.	
			9	T1_OUT0 ctimer 1 match[0] output.	
			0xA	T2_OUT0 ctimer 2 match[0] output.	
			0xB	T3_OUT0 ctimer 3 match[0] output.	
			0xC	T4_OUT0 ctimer 4 match[0] output.	
			0xD	ADC_IRQ interrupt request from ADC.	
			0xE	GPIOINT_BMATCH.	
			0xF	USB0_FRAME_TOGGLE.	
			0x10	USB1_FRAME_TOGGLE.	
			0x11	COMP_OUTPUT output from analog comparator.	
			0x12	I2S_SHARED_SCK[0] output from I^2S pin sharing.	
			0x13	I2S_SHARED_SCK[1] output from I^2S pin sharing.	
			0x14	I2S_SHARED_WS[0] output from I^2S pin sharing.	
			0x15	I2S_SHARED_WS[1] output from I^2S pin sharing.	
			0x16	ARM_TXEV interrupt event from cpu0 or cpu1.	
			0x17	DEBUG_HALTED from cpu0 or cpu1.	
			0x18-0x1F	None.	
31:5				Reserved.	undefined

For functions selected from IOCON registers, see Section 15.5.4 “IOCON pin functions in relation to FUNC values”

18.6.2 Capture select registers for timers 0 to 4

For each of the 5 standard timers, numbered i = 0 to 4 there are 4 TIMERiCAPTSELj, with j = 0 to 3, each allowing selecting between 25 external or internal input sources.

The output of TIMER0CAPTSEL0 Input multiplexing register 0 selects the source for TIMER0 capture input 0. The output of TIMER0CAPTSEL1 Input multiplexing register 1 selects the source for TIMER0 capture input 1, and so forth up to TIMER4CAPTSEL3 Input multiplexing register 3, which selects the input for TIMER4 capture input 3.

Table 373. TIMERiCAPTSELj Input multiplexing registers i = 0:4, j = 0:3 (Offsets 0x020:0x02C, 0x040:0x04C, 0x060:0x06C, 0x1A0:0x1AC, 0x1C0:0x1CC)

Bit	Symbol	Value	Description	Reset value
4:0	CAPTSEL		Input number to TIMER1 capture inputs 0 to 4.	0x1F
		0x0	CT_INP0 function selected from IOCON register.	
		0x1	CT_INP1 function selected from IOCON register.	
		0x2	CT_INP2 function selected from IOCON register.	
		0x3	CT_INP3 function selected from IOCON register.	
		0x4	CT_INP4 function selected from IOCON register.	
		0x5	CT_INP5 function selected from IOCON register.	
		0x6	CT_INP6 function selected from IOCON register.	
		0x7	CT_INP7 function selected from IOCON register.	
		0x8	CT_INP8 function selected from IOCON register.	
		0x9	CT_INP9 function selected from IOCON register.	
		0xA	CT_INP10 function selected from IOCON register.	
		0xB	CT_INP11 function selected from IOCON register.	
		0xC	CT_INP12 function selected from IOCON register.	
		0xD	CT_INP13 function selected from IOCON register.	
		0xE	CT_INP14 function selected from IOCON register.	
		0xF	CT_INP15 function selected from IOCON register.	
		0x10	CT_INP16 function selected from IOCON register.	
		0x11	None	
		0x12	None	
		0x13	None	
		0x14	USB0_FRAME_TOGGLE.	
		0x15	USB1_FRAME_TOGGLE.	
		0x16	COMP_OUTPUT output from analog comparator.	
		0x17	I^2S_SHARED_WS[0] output from I^2S pin sharing.	
		0X18	I^2S_SHARED_WS[1] output from I^2S pin sharing.	
		0x19-0x1F	None.	
31:5	-	-	Reserved.	-

For functions selected from IOCON registers, see Section 15.5.4 “IOCON pin functions in relation to FUNC values”

18.6.3 Pin interrupt select registers

Each of these eight registers selects one pin from among ports 0 and 1 as the source of a pin interrupt or as the input to the pattern match engine. To select a pin for any of the 8 pin interrupts or pattern match engine inputs, write the GPIO port pin number as 0 to 31 for pins PIO0_0 to PIO0_31 to the INTPIN bits. Port 1 pins correspond to pin numbers 32 to 63. For example, setting INTPIN to 0x5 in PINTSEL0 selects pin PIO0_5 for pin interrupt 0. To determine the GPIO port pin number for a given device package, see the pin description table in the data sheet.

Each of the pin interrupts must be enabled in the NVIC, see Section 19.5.2 “Pattern match engine” before it becomes active.

To use the selected pins for pin interrupts or the pattern match engine, see Table 8.

Table 374. Pin interrupt select registers (PINTSEL[0:7], offsets [0x0C0:0x0DC])

Bit	Symbol	Description	Reset value
6:0	INTPIN	Pin number select for pin interrupt or pattern match engine input. For PIOx_y: INTPIN = (x * 32) + y. PIO0_0 to PIO1_31 correspond to numbers 0 to 63.	0x7F
31:7	-	Reserved.	-

18.6.4 Pin interrupt secure select registers

Each of these two registers selects one pin from port 0 as the source of a pin interrupt or as the input to the pattern match engine. To select a pin for any of the 2 pin interrupts or pattern match engine inputs, write the GPIO port pin number as 0 to 31 for pins PIO0_0 to PIO0_31 to the INTPIN bits. For example, setting INTPIN to 0x5 in PINTSEL0 selects pin PIO0_5 for pin interrupt 0. To determine the GPIO port pin number for a given device package, see the pin description table in the data sheet.

Each of the pin interrupts must be enabled in the NVIC, see Table 8 before it becomes active.

To use the selected pins for pin interrupts or the pattern match engine, see Section 19.5.2 “Pattern match engine”.

Table 375. Pin interrupt secure select registers (PINTSECSEL [0:1], offsets 0x1e0 and 0x1e4)

Bit	Symbol	Description	Reset value
5:0	INTPIN	Pin number select for pin interrupt secure or pattern match engine input. For PIO0_x: INTPIN = x. PIO0_0 to PIO0_31 correspond to numbers 0 to 31.	0x3F
31:6	-	Reserved.	-

18.6.5 DMA0 trigger input multiplexing registers 0 to 22

With the DMA trigger input multiplexing registers, one trigger input can be selected for each of the DMA channels from the potential internal sources. By default, none of the triggers are selected.

Table 376. DMA0 trigger Input multiplexing registers (DMA0_ITRIG_INMUX[0:22], offsets [0x0E0:0x138])

Bit	Symbol	Value	Description	Reset value
4:0	INP		Trigger input number (decimal value) for DMA channel n (n = 0 to 22).	0x1F
		0	Pin interrupt 0.	
		1	Pin interrupt 1.	
		2	Pin interrupt 2.	
		3	Pin interrupt 3.	
		4	Timer CTIMER0 Match 0.	
		5	Timer CTIMER0 Match 1.	
		6	Timer CTIMER1 Match 0.	
		7	Timer CTIMER1 Match 1.	
		8	Timer CTIMER2 Match 0.	
		9	Timer CTIMER2 Match 1.	
		10	Timer CTIMER3 Match 0.	
		11	Timer CTIMER3 Match 1.	
		12	Timer CTIMER4 Match 0.	
		13	Timer CTIMER4 Match 1.	
		14	Comparator 0 output.	
		15	DMA output trigger 0.	
		16	DMA output trigger 1.	
		17	DMA output trigger 2.	
		18	DMA output trigger 3.	
		19	SCT0 DMA request 0.	
		20	SCT0 DMA request 1.	
		21	Hash-Crypt output DMA.	
31:5	-	-	Reserved.	-

18.6.6 DMA0 output trigger feedback multiplexing registers 0 to 3

This register provides a multiplexer for inputs 15 to 18 of each DMA trigger input multiplexing register DMA0_ITRIG_INMUX. These inputs can be selected from among the trigger outputs generated by each DMA channel. By default, none of the triggers are selected.

Table 377. DMA0 output trigger feedback multiplexing registers (DMA0_OTRIG_INMUX[0:3], offset [0x160:0x16C])

Bit	Symbol	Description	Reset value
4:0	INP	DMA trigger output number (decimal value) for DMA0 channel n (n = 0 to 22).	0x1F
31:5	-	Reserved.	-

18.6.7 DMA1 trigger input multiplexing registers 0 to 9

With the DMA trigger input multiplexing registers, one trigger input can be selected for each of the DMA channels from the potential internal sources. By default, none of the triggers are selected.

Table 378. DMA1 trigger Input multiplexing registers (DMA1_ITRIG_INMUX[0:9], offsets [0x200:0x224])

Bit	Symbol	Value	Description	Reset value
3:0	INP		Trigger input number (decimal value) for DMA1 channel n (n = 0 to 9).	0x1F
		0	Pin interrupt 0.	
		1	Pin interrupt 1.	
		2	Pin interrupt 2.	
		3	Pin interrupt 3.	
		4	Timer CTIMER0 Match 0.	
		5	Timer CTIMER0 Match 1.	
		6	Timer CTIMER2 Match 0.	
		7	Timer CTIMER4 Match 0.	
		8	DMA output trigger 0.	
		9	DMA output trigger 1.	
		10	DMA output trigger 2.	
		11	DMA output trigger 3.	
		12	SCT0 DMA request 0.	
		13	SCT0 DMA request 1.	
		14	Hash-Crypt output DMA.	
31:4	-	-	Reserved.	-

18.6.8 DMA1 output trigger feedback multiplexing registers 0 to 3

This register provides a multiplexer for inputs 8 to 11 of each DMA trigger input multiplexing register DMA1_ITRIG_INMUX. These inputs can be selected from among the trigger outputs generated by each DMA channel. By default, none of the triggers are selected.

Table 379. DMA1 output trigger feedback multiplexing registers (DMA1_OTRIG_INMUX[0:3], offset [0x240:0x24C])

Bit	Symbol	Description	Reset value
3:0	INP	DMA trigger output number (decimal value) for DMA channel n (n = 0 to 9).	0xF
31:5		Reserved.	

18.6.9 Frequency measure function reference clock select register

This register selects a clock for the reference clock of the frequency measure function. By default, no clock is selected.

Also see:

- Section 11.3.1 “Measure the frequency of a clock signal”.
- Section 11.6.1 “Frequency measure function”.
- Section 11.5.3 “Frequency measure function control register ”.
- Section 18.6.10 “Frequency measure function target clock select register”.

Table 380. Frequency measure function frequency clock select register (FREQMEAS_REF, offset 0x180)

Bit	Symbol	Description	Reset value
4:0	CLKIN	Clock source number (decimal value) for frequency measure function reference clock: 0 = External 16-32 MHz crystal oscillator 1 = FRO 12 MHz oscillator (fro12m_clk) 2 = FRO 96 MHz oscillator (fro96m_clk) 3 = Watchdog oscillator (wdt_clk ⇔ FRO 1 MHz oscillator) 4 = 32 kHz RTC oscillator (32k_clk ⇔ crystal 32kHz oscillator or FRO 32 kHz oscillator) 5 = Main clock (main_clk) See Section 4.5.35 “Main clock source select register B” 6 = FREQME_GPIO_CLK_A 7 = FREQME_GPIO_CLK_B	0xF
31:5	-	Reserved.	-

18.6.10 Frequency measure function target clock select register

This register selects a clock for the target clock of the frequency measure function. By default, no clock is selected. See Section 11.6.1 “Frequency measure function”, Section 11.3.1 “Measure the frequency of a clock signal”, and Section 11.5.3 “Frequency measure function control register ” and Section 18.6.9 “Frequency measure function reference clock select register” for more details.

Table 381. Frequency measure function target clock select register (FREQMEAS_TARGET, offset 0x184)

Bit	Symbol	Description	Reset value
4:0	CLKIN	Clock source number (decimal value) for frequency measure function target clock: 0 = External 16-32 MHz crystal oscillator () 1 = FRO 12 MHz oscillator (fro12m_clk) 2 = FRO 96 MHz oscillator (fro96m_clk) 3 = Watchdog oscillator (wdt_clk ⇔ FRO 1 MHz oscillator) 4 = 32 kHz RTC oscillator (32k_clk ⇔ crystal 32kHz oscillator or FRO 32 kHz oscillator) 5 = Main clock (main_clk), See Section 4.5.35 “Main clock source select register B” 6 = FREQME_GPIO_CLK_A 7 = FREQME_GPIO_CLK_B	0xF
31:5		Reserved.	-

18.6.11 DMA security registers

DMA requests to each of the two DMA controllers are enabled separately so that the same request is routed to only one DMA controller. Inputs to DMA0 are enabled by the DMA0 request enable register.

18.6.11.1 DMA0 request enable register

Inputs to DMA0 are enabled by the DMA0 request enable register. For each bit in this register, a 0 means that DMA request input is disabled and 1 means that DMA request input is enabled.

Table 382. DMA0 request enable register (DMA0_REQ_ENA, offset = 0x740)

Bit	Symbol	Description	Reset value
0	REQ_ENA0	Hash-Crypt DMA request.	0x0
1	REQ_ENA1	Spare channel, no request connected.	-
2	REQ_ENA2	High Speed SPI (Flexcomm 8) RX.	0x0
3	REQ_ENA3	High Speed SPI (Flexcomm 8) TX.	0x0

Table 382. DMA0 request enable register (DMA0_REQ_ENA, offset = 0x740) *...continued*

Bit	Symbol	Description	Reset value
4	REQ_ENA4	Flexcomm Interface 0 RX / I2C Slave.	0x0
5	REQ_ENA5	Flexcomm Interface 0 TX / I2C Master.	0x0
6	REQ_ENA6	Flexcomm Interface 1 RX / I2C Slave.	0x0
7	REQ_ENA7	Flexcomm Interface 1 TX / I2C Master.	0x0
8	REQ_ENA8	Flexcomm Interface 3 RX / I2C Slave.	0x0
9	REQ_ENA9	Flexcomm Interface 3 TX / I2C Master.	0x0
10	REQ_ENA10	Flexcomm Interface 2 RX / I2C Slave.	0x0
11	REQ_ENA11	Flexcomm Interface 2 TX / I2C Master.	0x0
12	REQ_ENA12	Flexcomm Interface 4 RX / I2C Slave.	0x0
13	REQ_ENA13	Flexcomm Interface 4 TX / I2C Master.	0x0
14	REQ_ENA14	Flexcomm Interface 5 RX / I2C Slave.	0x0
15	REQ_ENA15	Flexcomm Interface 5 TX / I2C Master.	0x0
16	REQ_ENA16	Flexcomm Interface 6 RX / I2C Slave.	0x0
17	REQ_ENA17	Flexcomm Interface 6 TX / I2C Master.	0x0
18	REQ_ENA18	Flexcomm Interface 7 RX / I2C Slave.	0x0
19	REQ_ENA19	Flexcomm Interface 7 TX / I2C Master.	0x0
20	REQ_ENA20	Spare channel, no request connected.	-
21	REQ_ENA21	ADC0 FIFO 0.	0x0
22	REQ_ENA22	ADC0 FIFO 1.	0x0
31:23	-	Reserved.	-

18.6.11.2 DMA0 request enable set register

Writing a 1 to a bit position in DMA0_REQ_ENA_SET, sets the corresponding position in DMA0_REQ_ENA. This is a write-only register. For bit assignments, see Section 18.6.11.1 “DMA0 request enable register”.

Table 383. DMA0 request enable set register (DMA0_REQ_ENA_SET, offset = 0x748)

Bit	Symbol	Access	Description	Reset value
22:0	SET	WO	Writing ones to this register sets the corresponding bit or bits in the DMA0_REQ_ENA register, if they are implemented. Bits that do not correspond to defined bits in DMA0_REQ_ENA are reserved and only zeroes should be written to them.	0x0
31:23		WO	Reserved.	undefined

18.6.11.3 DMA0 request enable clear register

Writing a 1 to a bit position in DMA0_REQ_ENA_CLR, clears the corresponding position in DMA0_REQ_ENA. This is a write-only register. For bit assignments, see Section 18.6.11.1 “DMA0 request enable register”.

Table 384. DMA0 request enable clear register (DMA0_REQ_ENA_CLR, offset = 0x750)

Bit	Symbol	Access	Description	Reset value
22:0	CLR	WO	Writing ones to this register clears the corresponding bit or bits in the DMA0_REQ_ENA register, if they are implemented. Bits that do not correspond to defined bits in DMA0_REQ_ENA are reserved and only zeroes should be written to them.	0x0
31:23		WO	Reserved.	-

18.6.11.4 DMA1 request enable register

Inputs to DMA1 are enabled by the DMA1 request enable register. For each bit in this register, a 0 means that DMA request input is disabled and 1 means that DMA request input is enabled.

Table 385. DMA1 request enable register (DMA1_REQ_ENA, offset = 0x760)

Bit	Symbol	Description	Reset value
0	REQ_ENA0	Hash-Crypt input DMA request.	0x0
1	REQ_ENA1	Spare channel, no request connected.	-
2	REQ_ENA2	High Speed SPI (Flexcomm 8) RX.	0x0
3	REQ_ENA3	High Speed SPI (Flexcomm 8) TX.	0x0
4	REQ_ENA4	Flexcomm Interface 0 RX /I^2C Slave.	0x0
5	REQ_ENA5	Flexcomm Interface 0 TX / I^2C Master.	0x0
6	REQ_ENA6	Flexcomm Interface 1 RX /I^2C Slave.	0x0
7	REQ_ENA7	Flexcomm Interface 1 TX / I^2C Master.	0x0
8	REQ_ENA8	Flexcomm Interface 3 RX / I^2C Slave.	0x0
9	REQ_ENA9	Flexcomm Interface 3 TX / I^2C Master.	0x0
31:10	-	Reserved.	-

18.6.11.5 DMA1 request enable set register

Writing a 1 to a bit position in DMA1_REQ_ENA_SET, sets the corresponding position in DMA1_REQ_ENA. This is a write-only register. For bit assignments, see Section 18.6.11.4 “DMA1 request enable register”.

Table 386. DMA1 request enable set register (DMA1_REQ_ENA_SET, offset = 0x768)

Bit	Symbol	Access	Description	Reset value
9:0	SET	WO	Writing ones to this register sets the corresponding bit or bits in the DMA1_REQ_ENA register, if they are implemented. Bits that do not correspond to defined bits in DMA1_REQ_ENA are reserved and only zeroes should be written to them.	0x0
31:10		WO	Reserved.	-

18.6.11.6 DMA1 request enable clear register

Writing a 1 to a bit position in DMA1_REQ_ENA_CLR, clears the corresponding position in DMA1_REQ_ENA. This is a write-only register. For bit assignments, see Section 18.6.11.4 “DMA1 request enable register”.

Table 387. DMA1 request enable clear register (DMA1_REQ_ENA_CLR, offset = 0x770)

Bit	Symbol	Access	Description	Reset value
9:0	CLR	WO	Writing ones to this register clears the corresponding bit or bits in the DMA1_REQ_ENA register, if they are implemented. Bits that do not correspond to defined bits in DMA1_REQ_ENA are reserved and only zeroes should be written to them.	0x0
31:10		WO	Reserved.	-

18.6.11.7 DMA0 input trigger enable register

DMA triggers to each of the two DMA controllers are enabled separately so that the same trigger can be routed to only one DMA controller. Inputs to DMA0 are enabled by this register.

Table 388. DMA0 input trigger enable register (DMA0_ITRIG_ENA, offset = 0x780)

Bit	Symbol	Access	Description	Reset value
21:0	ITRIG_ENA	RW	Controls the 22 trigger inputs of DMA0. If bit i is ’1’ the DMA trigger input #i is enabled.	0x3FFFFF
31:22		WO	Reserved.	-

18.6.11.8 DMA0 input trigger enable set register

The DMA0 input trigger enable set register allows setting any combination of bits in the DMA0_ITRIG_ENA register.

Table 389. DMA0 input trigger enable set register (DMA0_ITRIG_ENA_SET, offset = 0x788)

Bit	Symbol	Access	Description	Reset value
21:0	SET	WO	Writing ones to this register sets the corresponding bit or bits in the DMA0_ITRIG_ENA register, if they are implemented. Bits that do not correspond to defined bits in DMA0_ITRIG_ENA are reserved and only zeroes should be written to them.	0x0
31:22		WO	Reserved.	-

18.6.11.9 DMA0 input trigger enable clear register

The DMA0 input trigger enable clear register allow clearing any combination of bits in the DMA0_ITRIG_ENA register.

Table 390. DMA0 input trigger enable clear register (DMA0_ITRIG_ENA_CLR, offset = 0x790)

Bit	Symbol	Access	Description	Reset value
21:0	CLR	WO	Writing ones to this register clears the corresponding bit or bits in the DMA0_ITRIG_ENA register, if they are implemented. Bits that do not correspond to defined bits in DMA0_ITRIG_ENA are reserved and only zeroes should be written to them.	0x0
31:22		WO	Reserved.	-

18.6.11.10 DMA1 input trigger enable register

DMA triggers to each of the two DMA controllers are enabled separately so that the same trigger can be routed to only one DMA controller. Inputs to DMA1 are enabled by this register.

Table 391. DMA1 input trigger enable register (DMA1_ITRIG_ENA, offset = 0x7A0)

Bit	Symbol	Access	Description	Reset value
14:0	ITRIG_ENA	RW	Controls the 15 trigger inputs of DMA1. If bit i is '1' the DMA trigger input #i is enabled.	0x7FFF
31:15		WO	Reserved.	undefined

18.6.11.11 DMA1 input trigger enable set register

The DMA1 input trigger enable set register allow setting any combination of bits in the DMA1_ITRIG_ENA register.

Table 392. DMA1 input trigger enable set register (DMA1_ITRIG_ENA_SET, offset = 0x7A8)

Bit	Symbol	Access	Description	Reset value
14:0	SET	WO	Writing ones to this register sets the corresponding bit or bits in the DMA1_ITRIG_ENA register, if they are implemented. Bits that do not correspond to defined bits in DMA1_ITRIG_ENA are reserved and only zeroes should be written to them.	0x0
31:15		WO	Reserved.	undefined

18.6.11.12 DMA1 input trigger enable clear register

The DMAC1 input trigger enable clear register allow clearing any combination of bits in the DMA1_ITRIG_ENA register.

Table 393. DMA1 input trigger enable clear register (DMA1_ITRIG_ENA_CLR, offset = 0x7B0)

Bit	Symbol	Access	Description	Reset value
14:0	CLR	WO	Writing ones to this register clears the corresponding bit or bits in the DMA1_ITRIG_ENA register, if they are implemented. Bits that do not correspond to defined bits in DMA1_ITRIG_ENA are reserved and only zeroes should be written to them.	0x0
31:15		WO	Reserved.	undefined

19.1 How to read this chapter

The pin interrupt generator and the pattern match engine are available on all LPC55S6x/LPC55S2x/LPC552x devices. PINT module uses standard GPIO functions as inputs.

19.2 Features

- Pin interrupts
  - Up to eight pins can be selected from all GPIO pins, on ports 0 and 1, as edge- or level-sensitive interrupt requests. Each request creates a separate interrupt in the NVIC.
  - Edge-sensitive interrupt pins can interrupt on rising or falling edges or both.
  - Level-sensitive interrupt pins can be HIGH- or LOW-active.
- Pattern match engine.
  - Up to eight pins can be selected from all digital pins on ports 0 and 1, 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 CPU0 and CPU1.
  - Pattern match can be used, in conjunction with software, to create complex state machines based on pin inputs.

19.3 Basic configuration

- Pin interrupts
  - Select up to eight external interrupt pins from all digital port pins on ports 0 and 1, in the Input Mux block, see Table 371 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 AHBCLKCTRL0 register, see Table 55.
  - In order to use the pin interrupts to wake up the part from deep-sleep mode, enable the pin interrupt wake-up feature using the low power API.

- Pattern match engine
  - Select up to eight external pins from all digital port pins on ports 0 and 1, in the Input mux block Table 371 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 AHBCLKCTRL0 register, see Table 55.
  - Each bit slice of the pattern match engine is assigned to one interrupt in the NVIC (interrupts #4 through #7 for pin interrupts 0 to 3, and 32 through 35 for pin interrupts 4 through 7).

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

Follow these steps to configure pins as pin interrupts:

1. Determine the pins that serve as pin interrupts on the LPC55S6x/LPC55S2x/LPC552x package. See the data sheet for determining the GPIO port pin number associated with the package pin.
2. For each pin interrupt, program the GPIO port pin number from ports 0 and 1 into one of the eight PINTSEL registers in the Input multiplexing block.

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

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

   See Section 18.6.3 “Pin interrupt select registers ” in the Input multiplexing block for the PINTSEL register.

Remark: The inputs to the pin interrupt select registers bypass the IOCON function selection. They do not have to be selected as GPIO in IOCON. Make sure that no analog function is selected on pins that are input to the pin interrupts.

19.4 Pin description

The inputs to the pin interrupt and pattern match engine are determined by the pin interrupt select registers in the Input multiplexing. See Section 18.6.3 “Pin interrupt select registers ”

19.5 General description

Pins with configurable functions can serve as external interrupts or inputs to the pattern match engine. Up to eight pins can be configured using the PINTSEL registers in the Input multiplexing block for these features.

19.5.1 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, seeTable 371 The external interrupt pins are connected to eight individual interrupts in the NVIC and are created based on rising or falling edges or on the input level on the pin.

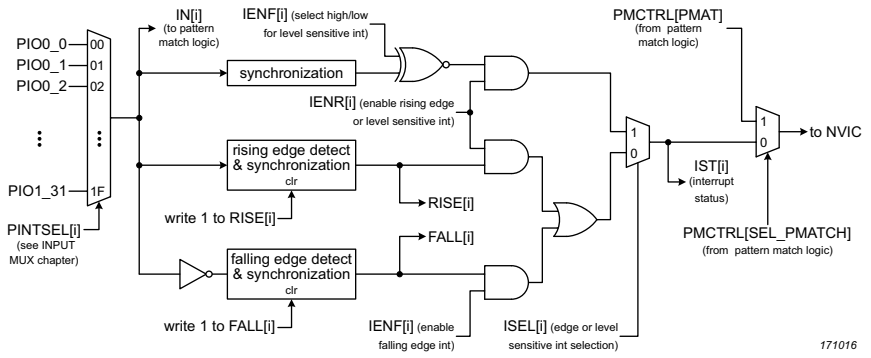


Fig 57. Pin interrupts

19.5.2 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 that monitors the selected input continuously and creates a HIGH output if the input qualifies as detected, that is as true. Several terms can be combined to a minterm and a pin interrupt is asserted when the minterm evaluates as true.

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 59 shows the details of the edge detection logic for each slice.

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

A time window can be created during which rising or falling edges can create a pin interrupt by combining a level detect with an event detect. See Section 19.7.3 “Pattern match engine edge detect examples” for details.

The connections between the pins and the pattern match engine are shown in Figure 58. All pins that are inputs to the pattern match engine can be GPIO port pins or other pin function depending on the IOCON configuration.

Remark: Note that the pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during deep-sleep mode.

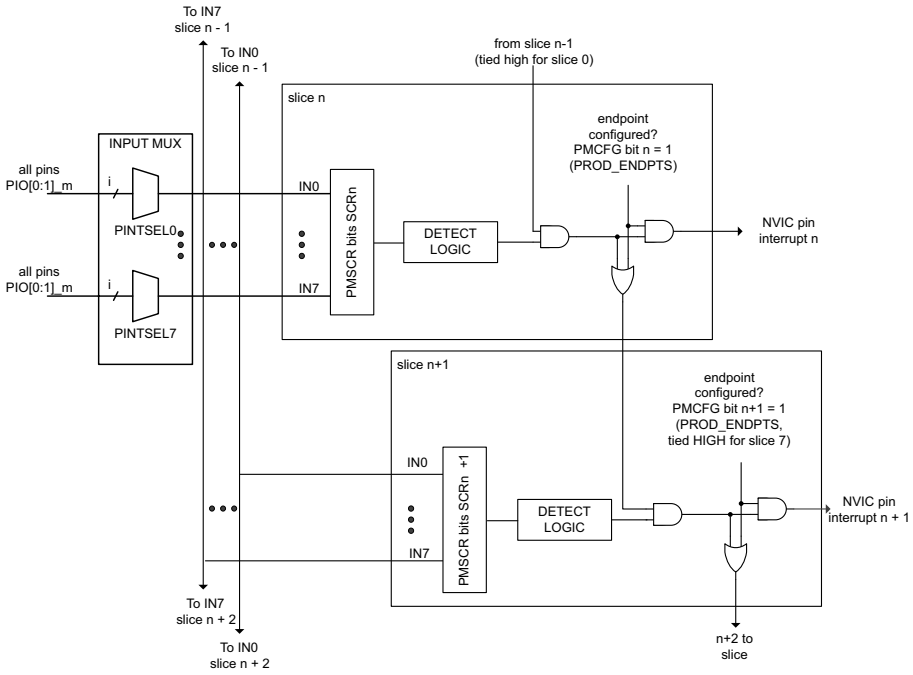


Fig 58. Pattern match engine connections.

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 cores (CPU0 and CPU1) when the entire 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 CPUs can still be enabled for pattern matches.

Remark: Pattern matching cannot be used to wake the part up from reduced power modes. Pin interrupts must be selected in order to use the GPIO for wake-up.

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 Figure 59 for bit slice drawing.

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.

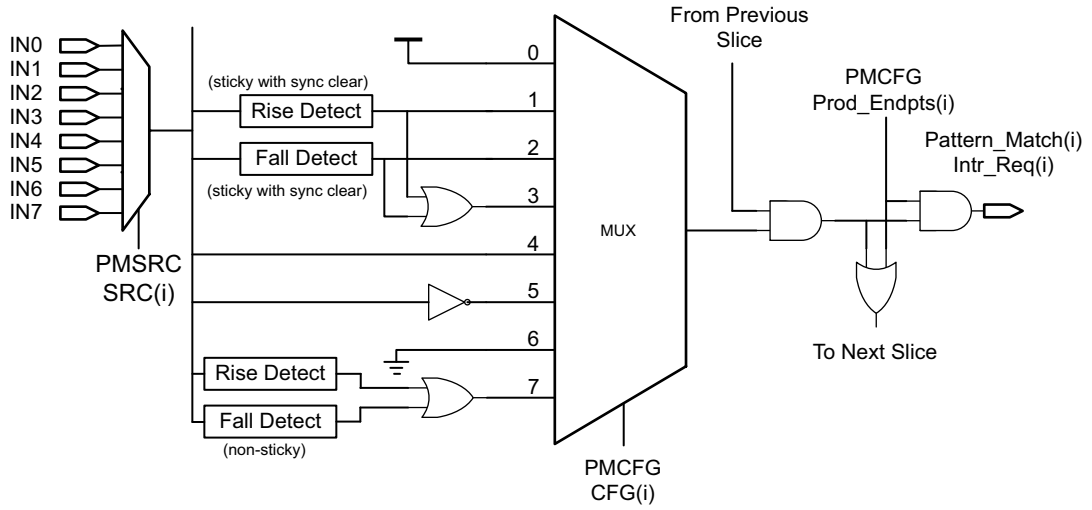


Fig 59. Pattern match bit slice with detect logic

19.5.2.1 Example

The following expression is specified through the registers PMSRC, see Table 406 and PMCFG Table 407:

```
IN0 AND NOT IN1 AND IN3 rising edge OR IN1 AND IN2 OR IN0 AND NOT IN3 AND NOT IN4
```

Each term in the boolean expression, IN0, NOT IN1, IN3 rising edge, etc., represents one bit slice of the pattern match engine.

- In the first AND function IN0 AND NOT IN1 AND IN3 rising edge, 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 will be asserted.
- In the second AND function IN1 AND 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 will be asserted.
- In the third AND function IN0 AND NOT IN3 AND NOT 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 will be asserted.

- The ORed result of all three AND functions asserts the RXEV request to the CPUs. That is, if any of the three terms are true, the output is asserted.

See Section 19.7.2 “Pattern match engine example” for more details.

19.6 Register description

Table 394. Register overview: Pin interrupts/pattern match engine (base address = 0x4000 4000)

Name	Access	Offset	Description	Reset value	Section
ISEL	R/W	0x000	Pin interrupt mode.	0	19.6.1
IENR	R/W	0x004	Pin interrupt level or rising edge interrupt enable.	0	19.6.2
SIENR	WO	0x008	Pin interrupt level or rising edge interrupt enable set.	NA	19.6.3
CIENR	WO	0x00C	Pin interrupt level or rising edge interrupt enable clear.	NA	19.6.4
IENF	R/W	0x010	Pin interrupt active level or falling edge interrupt enable.	0	19.6.5
SIENF	WO	0x014	Pin interrupt active level or falling edge interrupt set.	NA	19.6.6
CIENF	WO	0x018	Pin interrupt active level or falling edge interrupt clear.	NA	19.6.7
RISE	R/W	0x01C	Pin interrupt rising edge.	0	19.6.8
FALL	R/W	0x020	Pin interrupt falling edge.	0	19.6.9
IST	R/W	0x024	Pin interrupt status.	0	19.6.10
PMCTRL	R/W	0x028	Pattern match interrupt control.	0	19.6.11
PMSRC	R/W	0x02C	Pattern match interrupt bit-slice source.	0	19.6.12
PMCFG	R/W	0x030	Pattern match interrupt bit slice configuration.	0	19.6.13

19.6.1 Pin interrupt mode register

For each of the 8 pin interrupts selected in the PINTSELn registers, see Section 18.6.3 “Pin interrupt select registers ”. One bit in the ISEL register determines whether the interrupt is edge or level sensitive.

Table 395. Pin interrupt mode register (ISEL, offset = 0x000)

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. Read value is undefined, only zero should be written.	-	-

19.6.2 Pin interrupt level or rising edge interrupt enable register

For each of the 8 pin interrupts selected in the PINTSELn registers see Section 18.6.3 “Pin interrupt select registers ”. 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 396. Pin interrupt level or rising edge interrupt enable register (IENR, offset = 0x004)

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. Read value is undefined, only zero should be written.	-	-

19.6.3 Pin interrupt level or rising edge interrupt enable set register

For each of the 8 pin interrupts selected in the PINTSELn registers, see Section 18.6.3 "Pin interrupt select registers ". 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.

Table 397. Pin interrupt level or rising edge interrupt enable set register (SIENR, offset = 0x008)

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.	-	-

19.6.4 Pin interrupt level or rising edge interrupt clear register

For each of the 8 pin interrupts selected in the PINTSELn registers, see Section 18.6.3 "Pin interrupt select registers ". 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.

Table 398. Pin interrupt level or rising edge interrupt clear register (CIENR, offset = 0x00C)

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.	-	-

19.6.5 Pin interrupt active level or falling edge interrupt enable register

For each of the 8 pin interrupts selected in the PINTSELn registers. see Section 18.6.3 "Pin interrupt select registers ". One bit in the IENF register enables the falling edge interrupt or 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 399. Pin interrupt active level or falling edge interrupt enable register (IENF, offset = 0x010)

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 or set active interrupt level HIGH.	0	R/W
31:8	-	Reserved.	-	-

19.6.6 Pin interrupt active level or falling edge interrupt set register

For each of the 8 pin interrupts selected in the PINTSELn registers, see Section 18.6.3 "Pin interrupt select registers ". 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 400. Pin interrupt active level or falling edge interrupt set register (SIENF, offset = 0x014)

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.	-	-

19.6.7 Pin interrupt active level or falling edge interrupt clear register

For each of the 8 pin interrupts selected in the PINTSELn registers, see Section 18.6.3 "Pin interrupt select registers ". One bit in the CIENF register clears 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 401. Pin interrupt active level or falling edge interrupt clear register (CIENF, offset = 0x018)

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.	-	-

19.6.8 Pin interrupt rising edge register

This register contains ones for pin interrupts selected in the PINTSELn registers, see Section 18.6.3 “Pin interrupt select registers ” 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 402. Pin interrupt rising edge register (RISE, offset = 0x01C)

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.	-	-

19.6.9 Pin interrupt falling edge register

This register contains ones for pin interrupts selected in the PINTSELn registers, see Section 18.6.3 “Pin interrupt select registers ” 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 403. Pin interrupt falling edge register (FALL, offset = 0x020)

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.	-	-

19.6.10 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 404. Pin interrupt status register (IST, offset = 0x024)

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.	-	-

19.6.11 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 CPUs. 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.

Remark: Note that the pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below sleep mode..

Table 405. Pattern match interrupt control register (PMCTRL, offset = 0x028)

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 CPUs when the specified boolean expression evaluates to true.	0
		0	Disabled. RXEV output to the CPUs is disabled.	
		1	Enabled. RXEV output to the CPUs 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

19.6.12 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 INPUTMUX block. See Section 18.6.3 “Pin interrupt select registers ”. 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 406. Pattern match bit-slice source register (PMSRC, offset = 0x02C)

Bit	Symbol	Value	Description	Reset value
7:0	Reserved		Software should not write 1s to unused bits.	-
10:8	SRC0		Selects the input source for bit slice 0	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 0.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 0.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 0.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 0.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 0.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 0.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 0.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 0.	
13:11	SRC1		Selects the input source for bit slice 1	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 1.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 1.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 1.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 1.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 1.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 1.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 1.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 1.	
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 406. Pattern match bit-slice source register (PMSRC, offset = 0x02C) *…continued*

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.	
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.	
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 6.	

Table 406. Pattern match bit-slice source register (PMSRC, offset = 0x02C) *…continued*

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.	

19.6.13 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 407. Pattern match bit slice configuration register (PMCFG, offset = 0x030)

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 6 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.	-

Table 407. Pattern match bit slice configuration register (PMCFG, offset = 0x030) *…continued*

Bit	Symbol	Value	Description	Reset value
10:8	CFG0		Specifies the match contribution condition for bit slice 0.	0
		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.	0
		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 407. Pattern match bit slice configuration register (PMCFG, offset = 0x030) *…continued*

Bit	Symbol	Value	Description	Reset value
16:14	CFG2		Specifies the match contribution condition for bit slice 2.	0
		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.	0
		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 407. Pattern match bit slice configuration register (PMCFG, offset = 0x030) *…continued*

Bit	Symbol	Value	Description	Reset value
22:20	CFG4		Specifies the match contribution condition for bit slice 4.	0
		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.	0
		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 407. Pattern match bit slice configuration register (PMCFG, offset = 0x030) *…continued*

Bit	Symbol	Value	Description	Reset value
28:26	CFG6		Specifies the match contribution condition for bit slice 6.	0
		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.	0
		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.	

19.7 Functional description

19.7.1 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 408.

Table 408. 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.

19.7.2 Pattern match engine example

Suppose the desired oBolean 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, see Table 406.
  - Since bit slice 5 will be used to detect a sticky event on input 6, a 1 can be written 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, see Table 407.
  - 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.
  - PROD_ENDPTS = 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: 100 - high level on the selected input (input 1) for bit slice 0
  - CFG1: 100 - high level on the selected input (input 1) for bit slice 1
  - CFG2: 100 - 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: 100 - 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, see Table 405.
  - 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 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.

19.7.3 Pattern match engine edge detect examples

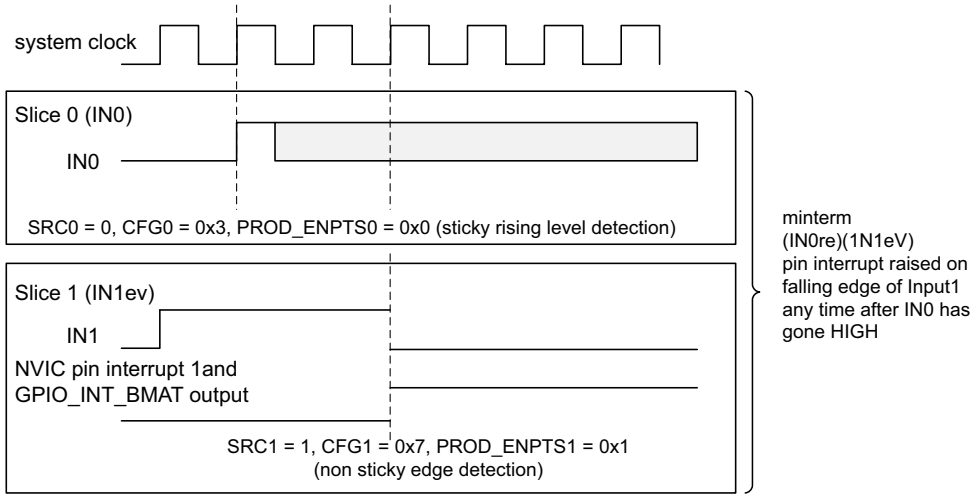


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 60. Pattern match engine examples: sticky edge detect

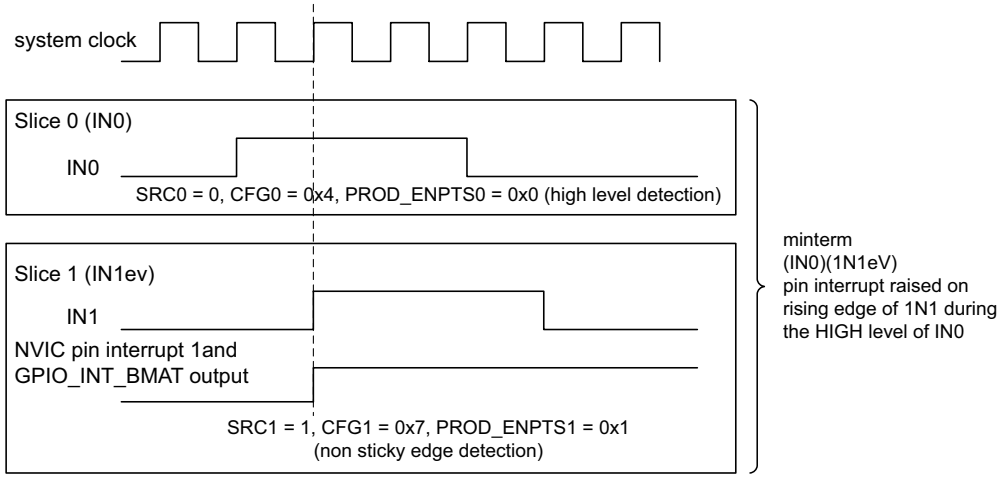


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 61. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true

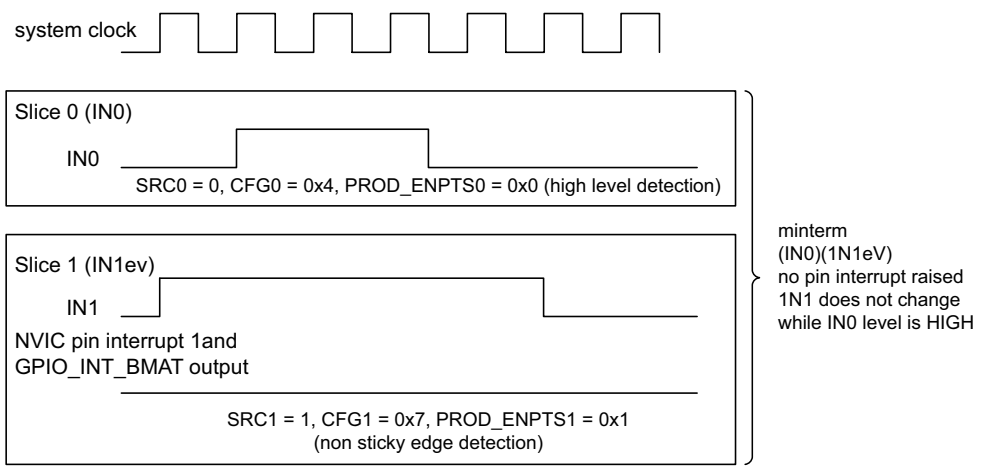


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 62. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false

20.1 How to read this chapter

The pin interrupt generator and the pattern match engine are available on all LPC55S6x/LPC55S2x/LPC552x devices. Secure PINT (or GPIO_INT_SEC) uses Secure GPIO functions but also any other pin input functions.

20.2 Features

- Pin interrupts
  - Up to two pins can be selected from all GPIO pins on port 0 for Secure PINT block, as edge-sensitive 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-active or LOW-active.
- Pattern match engine
  - Up to two pins can be selected from all digital pins on port 0 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 CPU0 and CPU1.
  - Pattern match can be used, in conjunction with software, to create complex state machines based on pin inputs.

20.3 Basic configuration

- Pin interrupts
  - In the input multiplexer block, select up to two external interrupt pins from all digital pins on port 0. See Table 371). 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 secure pin interrupt register block (GPIO_SEC_INT) in the AHBCLKCTRL2 register, see Table 57.
  - To use the pin interrupts to wake up the part from deep-sleep mode, enable the pin interrupt wake-up feature using low power API.
- Pattern match engine
  - Select up to two external pins from all digital port pins on ports 0 in the input mux block, see Table 371. 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 AHBCLKCTRL2 register, see Table 57.

- Each bit slice of the pattern match engine is assigned to one interrupt in the NVIC (interrupt #50 for pin interrupt 0 and #51 for pin interrupt 1).

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

Follow these steps to configure pins as pin interrupts:

1. Determine the pins that serve as pin interrupts on the LPC55S6x/LPC55S2x/LPC552x package. See the data sheet for determining the GPIO port pin number associated with the package pin.
2. For each pin interrupt, program the GPIO port pin number from port 0 into one of the two PINTSECSEL registers in the input multiplexer block.

   Remark: The port pin number serves to identify the pin to the PINTSECSEL registers. Any function, including GPIO, can be assigned to this pin via IOCON (not only P0_SEC(n) function).
3. Enable each pin interrupt in the NVIC.

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

   See Section 18.6.3 “Pin interrupt select registers ” in the input multiplexer block for the PINTSECSEL registers.

   Remark: The inputs to the pin interrupt select registers bypass the IOCON function selection. They do not have to be selected as P0_SEC(n) in IOCON. Make sure that no analog function is selected on pins that are input to the pin interrupts.

20.4 Pin description

The inputs to the pin interrupt and pattern match engine are determined by the pin interrupt secure select registers (PINTSECSELn) in the input multiplexer. See Section 18.6.3 “Pin interrupt select registers ”.

20.5 General description

Pins with configurable functions can serve as external interrupts or inputs to the pattern match engine. Up to two pins can be configured using the PINTSECSEL registers in the input multiplexer block for these features.

20.5.1 Pin interrupts

For the secure PINT block, from all available GPIO pins of P0 port, up to two pins can be selected in the system control block to serve as external interrupt pins, see Table 371. The external interrupt pins are connected to two individual interrupts in the NVIC and are created based on rising or falling edges or on the input level on the pin.

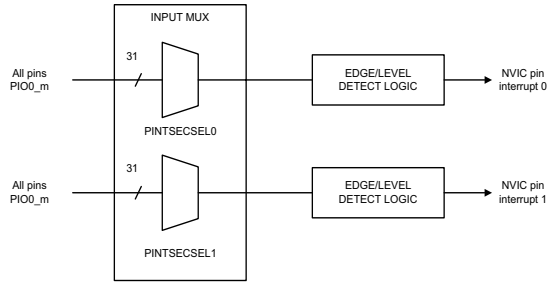


Fig 63. Secure pin interrupt connections.

20.5.2 Pattern match engine

The pattern match feature allows complex boolean expressions to be constructed from the same set of two 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 that monitors the selected input continuously and creates a HIGH output if the input qualifies as detected, that is as true. Several terms can be combined to a minterm and a pin interrupt is asserted when the minterm evaluates as true.

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 65 shows the details of the edge detection logic for each slice in Secure PINT block.

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

A time window can be created during which rising or falling edges can create a pin interrupt by combining a level detect with an event detect. See Section 20.7.3 “Pattern match engine edge detect examples” for details.

The connections between the pins and the pattern match engine are shown in Figure 64. All pins that are inputs to the pattern match engine can be GPIO port pins or other pin function depending on the IOCON configuration.

Remark: Note that the pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during deep-sleep mode.

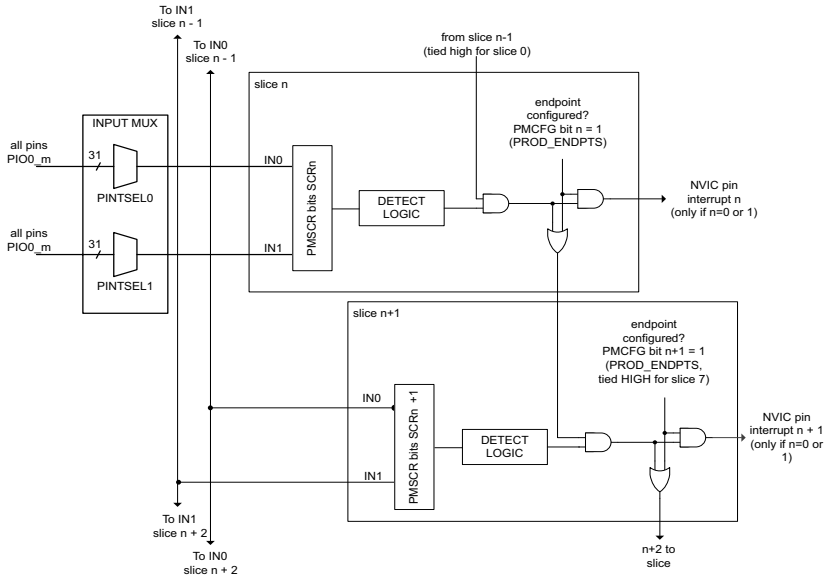


Fig 64. Pattern match engine connections

The pattern match logic continuously monitors the 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 individual minterm 0 and 1.

In addition, the pattern match module can be enabled to generate a Receive Event (RXEV) output to the ARM cores (CPU0 and CPU1) when the entire boolean expression is true (i.e. when any minterm is matched).

The pattern match function utilizes the same two 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 CPUs can still be enabled for pattern matches.

Remark: Pattern matching cannot be used to wake the part up from reduced power modes. Pin interrupts must be selected in order to use the GPIO for wake-up.

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. For bit slices 0 and 1 only, the interrupt request associated with the last bit slice (either #0 or #1) for a particular minterm will be asserted whenever that minterm is matched. See bit slice drawing Figure 65.

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.

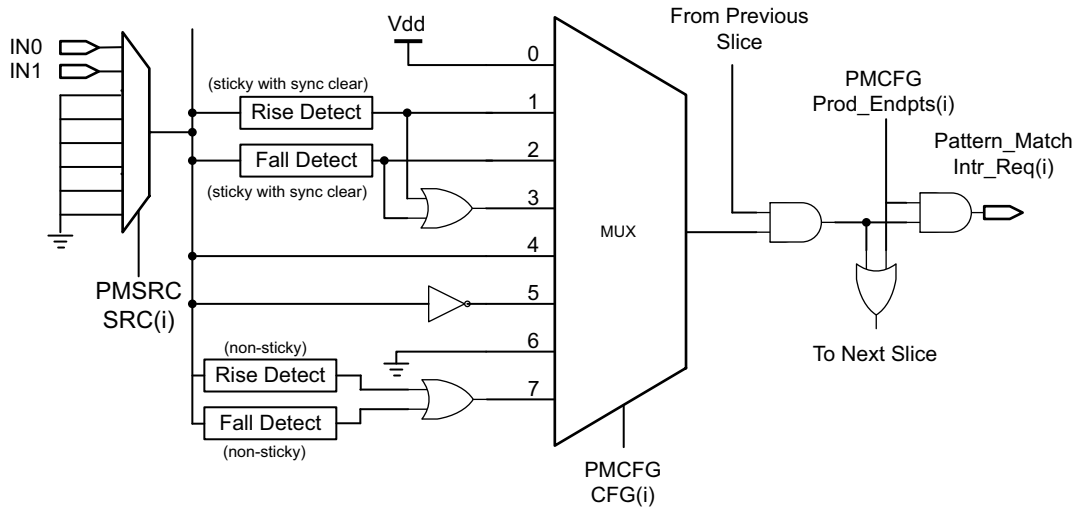


Fig 65. Secure pattern match bit slice with detect logic

20.5.2.1 Example

Assume the expression: (IN0)v~(IN1) + (IN0)^(IN1) is specified through the registers PMSRC Table 421 and PMCFG Table 422. Each term in the boolean expression, (IN0)v, ~(IN1), (IN0)^, etc., represents one bit slice of the pattern match engine.

- In the first minterm (IN0)v~(IN1), bit slice 0 monitors for a falling-edge on input (IN0) and bit slice 1 monitors for a low level on input (IN1). If this combination is detected, that is if both terms are true, the interrupt associated with bit slice 1 will be asserted.
- In the second minterm (IN0)^(IN1), bit slice 2 monitors input (IN0) for a rising-edge, bit slice 3 monitors input (IN1) for a high level. If this combination is detected, the interrupt associated with bit slice 3 will be asserted but will not be propagated to the NVIC since only slices 0 and 1 have their interrupt connected.
- The ORed result of both minterms asserts the RXEV request to the CPU. That is, if any of the three terms are true, the output is asserted.

Related links: Section 20.7.2 “Pattern match engine example”

20.6 Register description

Table 409. Register overview: Pin interrupts/pattern match engine (base address = 0x4000 5000)

Name	Access	Offset	Description	Reset value	Section
ISEL	R/W	0x000	Pin interrupt mode.	0	20.6.1
IENR	R/W	0x004	Pin interrupt level or rising edge interrupt enable.	0	20.6.2
SIENR	WO	0x008	Pin interrupt level or rising edge interrupt enable set.	NA	20.6.3
CIENR	WO	0x00C	Pin interrupt level or rising edge interrupt enable clear.	NA	20.6.4
IENF	R/W	0x010	Pin interrupt active level or falling edge interrupt enable.	0	20.6.5
SIENF	WO	0x014	Pin interrupt active level or falling edge interrupt set.	NA	20.6.6
CIENF	WO	0x018	Pin interrupt active level or falling edge interrupt clear.	NA	20.6.7
RISE	R/W	0x01C	Pin interrupt rising edge.	0	20.6.8
FALL	R/W	0x020	Pin interrupt falling edge.	0	20.6.9
IST	R/W	0x024	Pin interrupt status.	0	20.6.10
PMCTRL	R/W	0x028	Pattern match interrupt control.	0	20.6.11
PMSRC	R/W	0x02C	Pattern match interrupt bit-slice source.	0	20.6.12
PMCFG	R/W	0x030	Pattern match interrupt bit slice configuration.	0	20.6.13

20.6.1 Pin interrupt mode register

In Secure PINT block, for each of the two pin interrupts selected in the PINTSECSELn registers, see Section 18.6.4, one bit in the ISEL register determines whether the interrupt is edge-sensitive or level-sensitive.

Table 410. Pin interrupt mode register (ISEL, offset = 0x000)

Bit	Symbol	Description	Reset value	Access
1:0	PMODE	Selects the interrupt mode for each pin interrupt. Bit n configures the pin interrupt selected in PINTSECSELn. 0 = Edge-sensitive 1 = Level-sensitive	0	R/W
31:2	-	Reserved. Read value is undefined, only zero should be written.	-	-

20.6.2 Pin interrupt level or rising edge interrupt enable register

For each of the two pin interrupts selected in the PINTSECSELn registers, see Section 18.6.4, 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 411. Pin interrupt level or rising edge interrupt enable register (IENR, offset = 0x004)

Bit	Symbol	Description	Reset value	Access
1:0	ENRL	Enables the rising edge or level interrupt for each pin interrupt. Bit n configures the pin interrupt selected in PINTSECSELn. 0 = Disable rising edge or level interrupt. 1 = Enable rising edge or level interrupt.	0	R/W
31:2	-	Reserved. Read value is undefined, only zero should be written.	-	-

20.6.3 Pin interrupt level or rising edge interrupt enable set register

For each of the two pin interrupts selected in the PINTSECSELn registers, see Section 18.6.4, 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.

Table 412. Pin interrupt level or rising edge interrupt enable set register (SIENR, offset = 0x008)

Bit	Symbol	Description	Reset value	Access
1:0	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
31:2	-	Reserved.	-	-

20.6.4 Pin interrupt level or rising edge interrupt clear register

For each of the two pin interrupts selected in the PINTSECSELn registers, see Section 18.6.4, 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.

Table 413. Pin interrupt level or rising edge interrupt clear register (CIENR, offset = 0x00C)

Bit	Symbol	Description	Reset value	Access
1:0	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
31:2	-	Reserved.	-	-

20.6.5 Pin interrupt active level or falling edge interrupt enable register

For each of the two pin interrupts selected in the PINTSECSELn registers, see Section 18.6.4, one bit in the IENF register enables the falling edge interrupt or 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 414. Pin interrupt active level or falling edge interrupt enable register (IENF, offset = 0x010)

Bit	Symbol	Description	Reset value	Access
1:0	ENAF	Enables the falling edge or configures the active level interrupt for each pin interrupt. Bit n configures the pin interrupt selected in PINTSECSELn. 0 = Disable falling edge interrupt or set active interrupt level LOW. 1 = Enable falling edge interrupt or set active interrupt level HIGH.	0	R/W
31:2	-	Reserved.	-	-

20.6.6 Pin interrupt active level or falling edge interrupt set register

For each of the two pin interrupts selected in the PINTSECSELn registers, see Section 18.6.4, 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 415. Pin interrupt active level or falling edge interrupt set register (SIENF, offset = 0x014)

Bit	Symbol	Description	Reset value	Access
1:0	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
31:2	-	Reserved.	-	-

20.6.7 Pin interrupt active level or falling edge interrupt clear register

For each of the two pin interrupts selected in the PINTSECSELn registers, see Section 18.6.4, one bit in the CIENF register clears 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 416. Pin interrupt active level or falling edge interrupt clear register (CIENF, offset = 0x018)

Bit	Symbol	Description	Reset value	Access
1:0	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
31:2	-	Reserved.	-	-

20.6.8 Pin interrupt rising edge register

This register contains ones for pin interrupts selected in the PINTSECSELn registers, see Section 18.6.4 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 PINTSECSELn registers, regardless of whether they are interrupt-enabled.

Table 417. Pin interrupt rising edge register (RISE, offset = 0x01C)

Bit	Symbol	Description	Reset value	Access
1:0	RDET	Rising edge detect. Bit n detects the rising edge of the pin selected in PINTSECSELn. 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
31:2	-	Reserved.	-	-

20.6.9 Pin interrupt falling edge register

In Secure PINT block, this register contains ones for pin interrupts selected in the PINTSECSELn registers, see Section 18.6.4 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 PINTSECSELn registers, regardless of whether they are interrupt-enabled.

Table 418. Pin interrupt falling edge register (FALL, offset = 0x020)

Bit	Symbol	Description	Reset value	Access
1:0	FDET	Falling edge detect. Bit n detects the falling edge of the pin selected in PINTSECSELn. 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
31:2	-	Reserved.	-	-

20.6.10 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 419. Pin interrupt status register (IST, offset = 0x024)

Bit	Symbol	Description	Reset value	Access
1:0	PSTAT	Pin interrupt status. Bit n returns the status, clears the edge interrupt, or inverts the active level of the pin selected in PINTSECSELn. 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
31:2	-	Reserved.	-	-

20.6.11 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 CPUs. 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.

Remark: note that the pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below sleep mode.

Table 420. Pattern match interrupt control register (PMCTRL, offset = 0x028)

Bit	Symbol	Value	Description	Reset value
0	SEL_PMATCH		Specifies whether the 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 CPUs when the specified boolean expression evaluates to true.	0
		0	Disabled. RXEV output to the CPUs are disabled.	
		1	Enabled. RXEV output to the CPUs are 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

20.6.12 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 two inputs is selected in the pin interrupt secure select registers in the INPUTMUX block, see Section 18.6.4. Input 0 corresponds to the pin selected in the PINTSECSEL0 register and input 1 corresponds to the pin selected in the PINTSECSEL1 register.

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 421. Pattern match bit-slice source register (PMSRC, offset = 0x02C)

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 to 0x7	Input 2 to 7. Tied to ‘0’ as the source to bit slice 0.	
13:11	SRC1		Selects the input source for bit slice 1.	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 1.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 1.	
		0x2 to 0x7	Input 2 to 7. Tied to ‘0’ 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 to 0x7	Input 2 to 7. Tied to ‘0’ as the source to bit slice 2.	
19:17	SRC3		Selects the input source for bit slice 3.	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 3.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 3.	
		0x2 to 0x7	Input 2 to 7. Tied to ‘0’ 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 to 0x7	Input 2 to 7. Tied to ‘0’ as the source to bit slice 4.	

Table 421. Pattern match bit-slice source register (PMSRC, offset = 0x02C) *…continued*

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 to 0x7	Input 2 to 7. Tied to '0' 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 to 0x7	Input 2 to 7. Tied to '0' as the source to bit slice 6.	
31:29	SRC7		Selects the input source for bit slice 7.	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 7.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 7.	
		0x2 to 0x7	Input 2 to 7. Tied to '0' as the source to bit slice 7.	

20.6.13 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).

Remark: Only 2 interrupt requests (from slices 0 and 1) are driven to the NVIC. Interrupt requests from other slices have no effect. However, slices 2 to 7 can be used to generate RXEV output to the CPU and pattern_match trigger.

Table 422. Pattern match bit slice configuration register (PMCFG, offset = 0x030)

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). Its output interrupt is raised if the minterm evaluates as true but it is not connected to the NVIC.	
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). Its output interrupt is raised if the minterm evaluates as true but it is not connected to the NVIC.	
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). Its output interrupt is raised if the minterm evaluates as true but it is not connected to the NVIC.	
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). Its output interrupt is raised if the minterm evaluates as true but it is not connected to the NVIC.	
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). Its output interrupt is raised if the minterm evaluates as true but it is not connected to the NVIC.	
7		-	Reserved. Bit slice 7 is automatically considered a product end point.	-

Table 422. Pattern match bit slice configuration register (PMCFG, offset = 0x030) *…continued*

Bit	Symbol	Value	Description	Reset value
10:8	CFG0		Specifies the match contribution condition for bit slice 0.	0
		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.	0
		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 422. Pattern match bit slice configuration register (PMCFG, offset = 0x030) *…continued*

Bit	Symbol	Value	Description	Reset value
16:14	CFG2		Specifies the match contribution condition for bit slice 2.	0
		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.	0
		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 422. Pattern match bit slice configuration register (PMCFG, offset = 0x030) *…continued*

Bit	Symbol	Value	Description	Reset value
22:20	CFG4		Specifies the match contribution condition for bit slice 4.	0
		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.	0
		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 422. Pattern match bit slice configuration register (PMCFG, offset = 0x030) *…continued*

Bit	Symbol	Value	Description	Reset value
28:26	CFG6		Specifies the match contribution condition for bit slice 6.	0
		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.	0
		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.	

20.7 Functional description

20.7.1 Pin interrupts

In this interrupt facility, up to 8 pins are identified as interrupt sources by the Pin Interrupt Secure Select registers (PINTSECSEL0-1). All registers in the pin interrupt block contain 2 bits, corresponding to the pins called out by the PINTSECSEL0-1 registers. The ISEL register defines whether each interrupt pin is edge-sensitive 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 423.

Table 423. Pin interrupt registers for edge-sensitive 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.

20.7.2 Pattern match engine example

Suppose the desired Boolean pattern to be matched is:

(IN0) + (IN0 * IN1) + (~IN0 * IN1fe) + (IN0 * IN1ev) + (IN0re) with:

IN1fe = (sticky) falling-edge on input 1

IN1ev = (non-sticky) event (rising or falling edge) on input 1

IN0re = (sticky) rising edge on input 0

Since there are only two possible inputs, IN0 and IN1, the boolean pattern may show some redundancy but it is just given as an example.
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, see Table 421:
  - Since bit slice 4 will be used to detect a sticky event on input 1, a 1 can be written to the SRC4 bits to clear any pre-existing edge detects on bit slice 4.
  - SRC0: 000 - select input 0 for bit slice 0
  - SRC1: 000 - select input 0 for bit slice 1
  - SRC2: 001 - select input 1 for bit slice 2
  - SRC3: 000 - select input 0 for bit slice 3
  - SRC4: 001 - select input 1 for bit slice 4
  - SRC5: 000 - select input 0 for bit slice 5

- SRC6: 001 - select input 1 for bit slice 6
- SRC7: 000 - select input 0 for bit slice 7

- PMCFG register, see Table 422.
  - PROD_ENDPTS0 = 1
  - PROD_ENDPTS2 = 1
  - PROD_ENDPTS4 = 1
  - PROD_ENDPTS6 = 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.
  - PROD_ENDPTS = 1010101 - bit slices 0, 2, 4, 6, and 7 are product-term endpoints. (Bit slice 7 is an endpoint by default - no associated register bit).
  - CFG0: 100 - high level on the selected input (input 0) for bit slice 0
  - CFG1: 100 - high level on the selected input (input 0) for bit slice 1
  - CFG2: 100 - high level on the selected input (input 1) for bit slice 2
  - CFG3: 101 - low level on the selected input (input 0) for bit slice 3
  - CFG4: 010 - (sticky) falling edge on the selected input (input 1) for bit slice 4
  - CFG5: 100 - high level on the selected input (input 0) for bit slice 5
  - CFG6: 111- event (any edge, non-sticky) on the selected input (input 1) for bit slice 6
  - CFG7: 001 - (sticky) rising edge on the selected input (input 0) for bit slice 7
- PMCTRL register, see Table 420.
  - 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, 4, 6, and/or 7 will be respectively asserted in response to a match on the second, the fourth, the sixth, and/or the seventh product term but will not be used since not driven to the NVIC.
  - Bit1: Setting this bit will cause the RxEv signal to the CPUs 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, 4, 6 and/or 7 may be high if the corresponding product terms are currently matching.
  - The remaining bits will always be low.

20.7.3 Pattern match engine edge detect examples

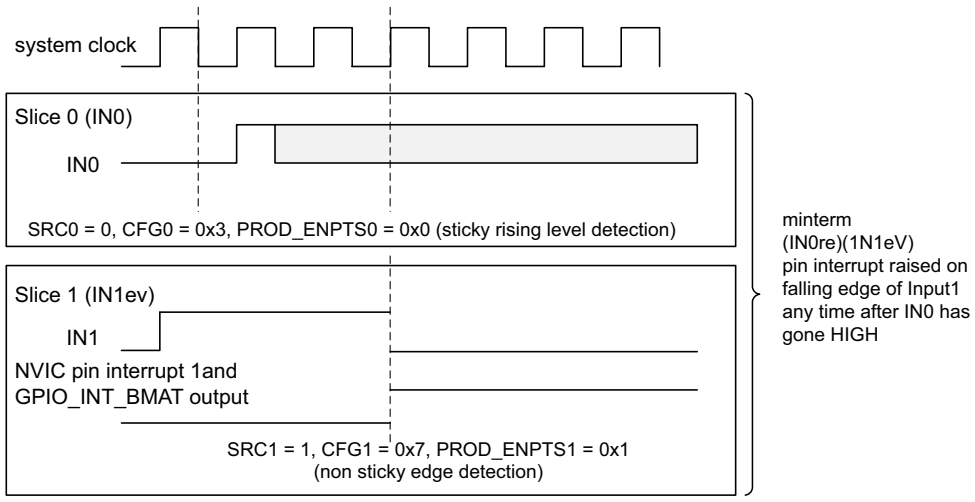


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 66. Pattern match engine examples: sticky edge detect

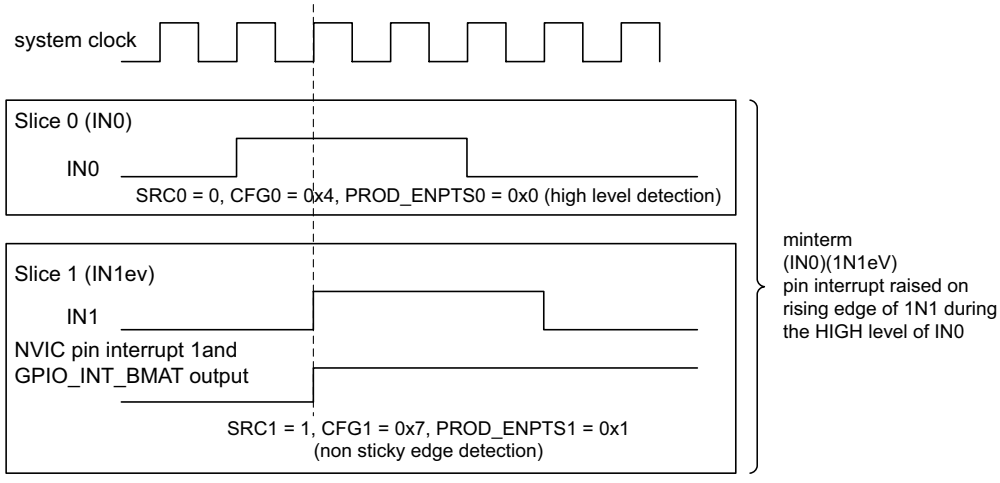


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 67. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true

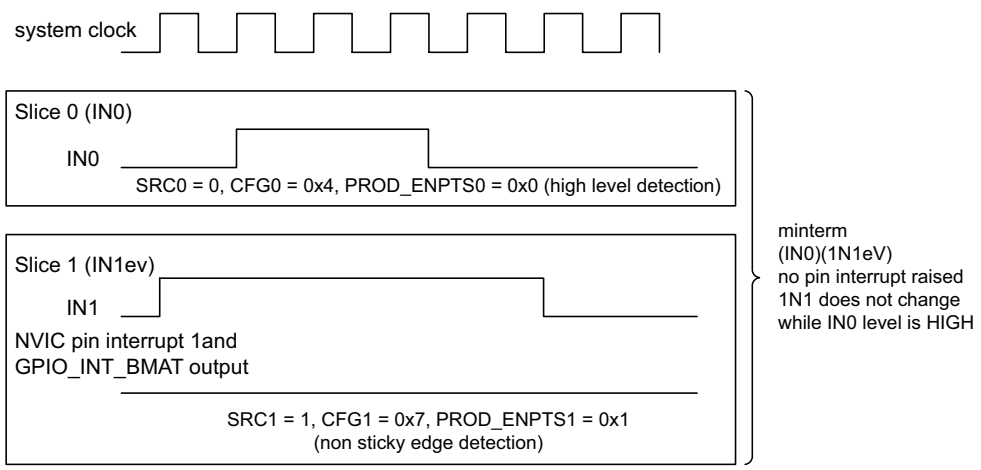


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 68. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false

21.1 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 mode, and power-down modes.

21.2 Basic configuration

For the group interrupt feature, enable the clock to both the GROUP0 and GROUP1 register interfaces in the AHBCLKCTRL0 register: GINT field. For sleep mode, the group interrupt wake-up feature is enabled in the CPUs NVIC. For deep-sleep and power-down low power modes, the group interrupt wake-up feature is enabled via the relevant low power API.

The pins can be configured as GPIO pins through IOCON, but they don’t have to be. The GINT block reads the input from the pin bypassing IOCON multiplexing. Make sure that no analog function is selected on pins that are input to the group interrupts. Selecting an analog function in IOCON disables the digital pad and the digital signal is tied to 0.

21.3 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 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.

21.4 Register description

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

Table 424. Register overview: GROUP0 interrupt (base address = 0x4000 2000 (GINT0) and 0x4000 3000 (GINT1))

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	GPIO grouped interrupt control.	0	21.4.1
PORT_POL0	R/W	0x020	GPIO grouped interrupt port 0 polarity.	0xFFFF FFFF	21.4.2
PORT_POL1	R/W	0x024	GPIO grouped interrupt port 1 polarity.	0xFFFF FFFF	21.4.2
PORT_ENA0	R/W	0x040	GPIO grouped interrupt port 0 enable.	0	21.4.3
PORT_ENA1	R/W	0x044	GPIO grouped interrupt port 1 enable.	0	21.4.3

21.4.1 Grouped interrupt control register

Table 425. GPIO grouped interrupt control register (CTRL, offset = 0x000)

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 request. No interrupt request is pending.	
		1	Request active. 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. Read value is undefined, only zero should be written.	0

21.4.2 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 426. GPIO grouped interrupt port polarity registers (PORT_POL[0:1], offset = 0x020 for PORT_POL0; 0x024 for PORT_POL1)

Bit	Symbol	Description	Reset value
31:0	POL	Configure pin polarity of port m pins for group interrupt. Bit n corresponds to pin PIOm_n of port m. 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

21.4.3 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 427. GPIO grouped interrupt port enable registers (PORT_ENA[0:1], offset = 0x040 for PORT_ENA0; 0x044 PORT_ENA1)

Bit	Symbol	Description	Reset value
31:0	ENA	Enable port 0 pin for group interrupt. Bit n corresponds to pin Pm_n of port m. 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.	1

21.5 Functional description

Any subset of the pins in each port can be selected to contribute to a common group interrupt (GINT) and can be enabled to wake the part up from deep-sleep and power-down modes.

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. 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 209.

22.1 How to read this chapter

The DMA controllers are available on all LPC55S6x/LPC55S2x/LPC552x devices.

22.2 Features

- DMA controller: Two instances of SDMA IP that the user can decide which one is secure or not.
- DMA0: 23 channels, with multiplexers for 22 trigger sources. Each Flexcomm Interface provides a DMA Rx and a DMA Tx request to the DMA controller. The ADC is connected to 2 different DMA request Channels. SCT and selected timers and pin interrupts may also be used as DMA triggers. In addition, four DMA triggers can be selected from among all of the DMA channel output triggers. SHA-2 and AES also provides DMA channel and trigger interface.
- DMA1: Ten channels with multiplexers for fifteen trigger sources.
- Priority is user selectable for each channel (up to eight priority levels).
- Continuous priority arbitration.
- Supports single transfers up to 1,024 words.
- Address increment options allow packing and/or unpacking data.

22.3 Basic configuration

Configure the DMA as follows:

- Use the AHBCLKCTRL0 register, seeTable 55 to enable the clock to the DMA0 registers interface.
- Use the AHBCLKCTRL2 register, see Table 57 to enable the clock to the DMA1 registers interface.
- Clear the DMA0 peripheral reset using the PRESETCTRL0 register, see Table 45.
- Clear the DMA1 peripheral reset using the PRESETCTRL2 register, seeTable 47.
- The DMA controller provides an interrupt to the NVIC, see Chapter 3 “LPC55S6x/LPC55S2x/LPC552x Nested Vectored Interrupt Controller (NVIC)”.
- Most peripherals that support DMA have at least one DMA request line associated with them. The related channel(s) should be set up according to the desired operation. DMA requests and triggers are described in detail in Section 22.5.1 “DMA requests and triggers”
- For peripherals using DMA requests, DMA operation must be triggered before any transfer occurs. It can be done by software, or can optionally be signalled by one of 15 hardware triggers, through the input multiplexers registers DMA_ITRIG_INMUX[0:22]. DMA requests and triggers are described in detail in Section 22.5.1 “DMA requests and triggers”

- Trigger outputs may optionally cause other DMA channels to be triggered for more complex DMA functions. Trigger outputs are connected to DMA_OTRIG_INMUX [0:3] as inputs to DMA triggers.
- For details on the trigger input and output multiplexing, see Section 18.5.4 “DMA trigger input multiplexing”.

SDMA block that interfaces to peripherals that support DMA requests, and to some additional peripherals that can generate DMA triggers.

Each Flexcomm Interface provides a DMA Rx and a DMA Tx request to the DMA controller. SCT and selected timers and pin interrupts may also be used as DMA triggers. In addition, four DMA triggers can be selected from among all of the DMA channel output triggers. SHA-2 and AES also provides DMA channel and trigger interface.

If enabled by the currently active security profile, the DMAC can access all on-chip RAM and flash memories, all AHB peripherals (potentially excepting those that do not support DMA or contain their own DMA engine), and all APB peripherals.

In case of TZ enabled device to support DMA operation for secure thread and non-secure thread two DMAs will be made available on this device. One DMA will be non-secure mode. Other DMA can be programmed by ROMCode to be secure DMA. Number of available channels and trigger MUX on second DMA is reduced. Also, to protect Secure DMA from insecure request or triggers a special masking mechanism is implemented, enabling DMA access for a particular device only if mask is disabled securely.

22.4 Pin description

The DMA controller has no direct pin connections. However, some DMA triggers can be associated with pin functions. See Section 22.5.1.2 “Hardware triggers”

22.5 General description

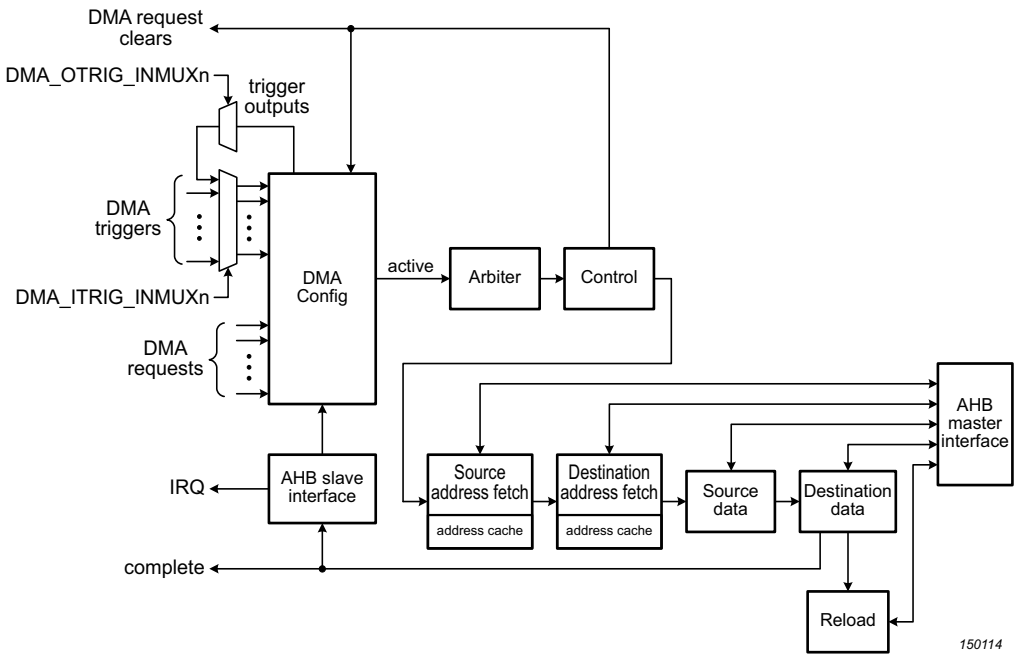


Fig 69. DMA block diagram

22.5.1 DMA requests and triggers

In general, DMA requests are intended to pace transfers to match what the peripheral (including its FIFO if it has one) can do. For example, the USART will issue a transmit DMA request when its transmit FIFO is not full, and a receive DMA request when its receive FIFO is not empty. DMA requests are summarized in Table 428.

Triggers start the transfer. In typical cases, only a software trigger will probably be used. Other possibilities are provided for, such as starting a DMA transfer when certain timer or pin related events occur. Those transfers would usually still be paced by a peripheral DMA request if a peripheral is involved in the transfer. Note: that no DMA activity will take place for any particular DMA channel unless that channel has been triggered, either by software or hardware. DMA triggers are summarized in Table 428.

DMA operations with ADC channels require special attention. When DMA operations are enabled in ADC registers (FWMDE0 = 1 in DE register, and/or FWMDE =1 in DE register), the ADC FIFO watermark level must be set to a minimum value of 2. See ADC registers (FWMARK bit in FCTRL0 register, and FWMARK bit in FCTRL1) in Chapter 39 “LPC55S6x/LPC55S2x/LPC552x 16-bit ADC controller (ADC)”.

Once triggered by software or hardware, a DMA operation on a specific channel is initiated by a DMA request if it is enabled for that channel.

A DMA channel using a trigger can respond by moving data from any memory address to any other memory address. This can include fixed peripheral data registers, or incrementing through RAM buffers. The amount of data moved by a single trigger event

can range from a single transfer to many transfers. A transfer that is started by a trigger can still be paced using the channel's DMA request. This allows sending a string to a serial peripheral, for instance, without overrunning the peripheral's transmit buffer.

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

22.5.1.1 DMA requests

DMA requests are directly connected to the peripherals. Each channel supports one DMA request line and one trigger input. Some DMA requests allow a selection of requests sources. DMA triggers are selected from many possible input sources. The requests and trigger MUXs for DMA controller 0 and 1 are shown in Table 428 and Table 429.

Table 428. DMA0 requests and trigger multiplexers

DMA channel	Request input	DMA trigger mux
0	Hash-Crypt DMA request	DMA0_ITRIG_INMUX0
1	Spare channel, no request connected	DMA0_ITRIG_INMUX1
2	High Speed SPI (Flexcomm 8) RX	DMA0_ITRIG_INMUX2
3	High Speed SPI (Flexcomm 8) TX	DMA0_ITRIG_INMUX3
4	Flexcomm Interface 0 RX / I2C Slave [1]	DMA0_ITRIG_INMUX4
5	Flexcomm Interface 0 TX / I2C Master [1]	DMA0_ITRIG_INMUX5
6	Flexcomm Interface 1 RX / I2C Slave [1]	DMA0_ITRIG_INMUX6
7	Flexcomm Interface 1 TX / I2C Master [1]	DMA0_ITRIG_INMUX7
8	Flexcomm Interface 3 RX / I2C Slave [1]	DMA0_ITRIG_INMUX8
9	Flexcomm Interface 3 TX / I2C Master [1]	DMA0_ITRIG_INMUX9
10	Flexcomm Interface 2 RX / I2C Slave [1]	DMA0_ITRIG_INMUX10
11	Flexcomm Interface 2 TX / I2C Master [1]	DMA0_ITRIG_INMUX11
12	Flexcomm Interface 4 RX / I2C Slave [1]	DMA0_ITRIG_INMUX12
13	Flexcomm Interface 4 TX / I2C Master [1]	DMA0_ITRIG_INMUX13
14	Flexcomm Interface 5 RX / I2C Slave [1]	DMA0_ITRIG_INMUX14
15	Flexcomm Interface 5 TX / I2C Master [1]	DMA0_ITRIG_INMUX15
16	Flexcomm Interface 6 RX / I2C Slave [1]	DMA0_ITRIG_INMUX16
17	Flexcomm Interface 6 TX / I2C Master [1]	DMA0_ITRIG_INMUX17
18	Flexcomm Interface 7 RX / I2C Slave [1]	DMA0_ITRIG_INMUX18
19	Flexcomm Interface 7 TX / I2C Master [1]	DMA0_ITRIG_INMUX19
20	Spare channel, no request connected	DMA0_ITRIG_INMUX20
21	ADC0 FIFO 0	DMA0_ITRIG_INMUX21
22	ADC0 FIFO 1	DMA0_ITRIG_INMUX22

[1] See Section 22.5.1.1.1 below for information about DMA for the I^2C monitor function.

Table 429. DMA1 requests and trigger multiplexers

DMA channel	Request input	DMA trigger mux
0	Hash-Crypt input DMA request	DMA1_ITRIG_INMUX0
1	Spare channel, no request connected	DMA1_ITRIG_INMUX1
2	High Speed SPI (Flexcomm 8) RX	DMA1_ITRIG_INMUX2
3	High Speed SPI (Flexcomm 8) TX	DMA1_ITRIG_INMUX3
4	Flexcomm Interface 0 RX / I2C Slave [1]	DMA1_ITRIG_INMUX4
5	Flexcomm Interface 0 TX / I2C Master [1]	DMA1_ITRIG_INMUX5
6	Flexcomm Interface 1 RX / I2C Slave [1]	DMA1_ITRIG_INMUX6
7	Flexcomm Interface 1 TX / I2C Master [1]	DMA1_ITRIG_INMUX7
8	Flexcomm Interface 3 RX / I2C Slave [1]	DMA1_ITRIG_INMUX8
9	Flexcomm Interface 3 TX / I2C Master [1]	DMA1_ITRIG_INMUX9

[1] See Section 22.5.1.1.1 below for information about DMA for the I²C monitor function.

22.5.1.1.1 DMA with I²C monitor mode

The I²C monitor function may be used with DMA if one of the channels related to the same Flexcomm Interface is available.

Table 430. DMA with the I²C

I²C Master DMA	I²C Slave DMA	I²C monitor DMA
Not enabled	-	If I²C Monitor DMA is enabled, it will use the DMA channel for the Master function other same Flexcomm Interface.
Enabled	Not enabled	If I²C Monitor is DMA enabled, it will use the DMA channel for the Slave function of the same Flexcomm Interface.
Enabled	Enabled	The I²C Monitor function cannot use DMA.

22.5.1.2 Hardware triggers

Each DMA channel can use one trigger that is independent of the request input for this channel. The trigger input is selected in the DMA_ITRIG_INMUX registers. There are 22 possible internal trigger sources for each DMA channel on DMA controller 0, and 15 possibilities for DMA controller 1. In addition, the DMA trigger output can be routed to the trigger input of another channel through the trigger input multiplexing. See Table 431 and Chapter 12 “LPC55xx Input multiplexing (INPUT MUX)”.

Table 431. DMA trigger sources

DMA trigger	DMA0 trigger input	DMA1 trigger input
0	Pin interrupt 0	Pin interrupt 0
1	Pin interrupt 1	Pin interrupt 1
2	Pin interrupt 2	Pin interrupt 2
3	Pin interrupt 3	Pin interrupt 3
4	Timer CTIMER0 Match 0	Timer CTIMER0 Match 0
5	Timer CTIMER0 Match 1	Timer CTIMER0 Match 1
6	Timer CTIMER1 Match 0	Timer CTIMER2 Match 0

Table 431. DMA trigger sources *…continued*

DMA trigger	DMA0 trigger input	DMA1 trigger input
7	Timer CTIMER1 Match 1	Timer CTIMER4 Match 0
8	Timer CTIMER2 Match 0	DMA output trigger 0
9	Timer CTIMER2 Match 1	DMA output trigger 1
10	Timer CTIMER3 Match 0	DMA output trigger 2
11	Timer CTIMER3 Match 1	DMA output trigger 3
12	Timer CTIMER4 Match 0	SCT0 DMA request 0
13	Timer CTIMER4 Match 1	SCT0 DMA request 1
14	Comparator 0 output	Hash-Crypt output DMA
15	DMA output trigger 0	NA
16	DMA output trigger 1	NA
17	DMA output trigger 2	NA
18	DMA output trigger 3	NA
19	SCT0 DMA request 0	NA
20	SCT0 DMA request 1	NA
21	Hash-Crypt output DMA	NA

22.5.1.3 Trigger operational detail

A trigger of some kind is always needed to start a transfer on a DMA channel. It 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.

22.5.1.4 Trigger output detail

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

22.5.2 DMA modes

The DMA controller does not 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 in Table 428, and supplying at least the channel descriptor, which is located somewhere in memory, typically in on-chip SRAM. See Section 22.6.3 “SRAM base address register”. The channel descriptor is shown in Table 432.

Table 432. 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 link to the next descriptor is used only if it is a linked transfer.

When a DMA transfer involves a fixed peripheral data register, such as, when moving data from memory to a peripheral or moving data from a peripheral to memory, the address used for SRCINC or DSTINC (whichever corresponds to the fixed peripheral data address) is the address of the peripheral data register. The memory address for such a transfer is based on the end (upper) address of the memory buffer. The value can be calculated from the starting address of the buffer and the length of the buffer, where the transfer increment is the value specified by SRCINC or DSTINC (whichever corresponds to the memory buffer):

Buffer ending address = buffer starting address + (XFERCOUNT * the transfer increment)

See Section 22.6.18 “Channel transfer configuration registers” for the description of SRCINC and DSTINC. Note that XFERCOUNT is defined as the actual count minus 1 and that is why it is not necessary to subtract 1 from the count in the equation above.

After the channel has 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 433. The difference is, 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 433. 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).

22.5.3 Single buffer

This generally applies to memory to memory moves, and peripheral DMA that occurs only occasionally and is set up for each transfer. For this kind of operation, only the initial channel descriptor shown in Table 434 is needed.

Table 434. Channel descriptor for a single transfer

Offset	Description
+ 0x0	Reserved.
+ 0x4	Source data end address.
+ 0x8	Destination 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.

22.5.4 Ping-Pong

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

A Ping-Pong transfer uses two buffers alternately. At any one time, one buffer is being loaded or unloaded by DMA operations. The other buffer has the opposite operation being handled by software, readying the buffer for use when the buffer currently being used by the DMA controller is full or empty. Table 435 shows an example of descriptors for ping-pong from a peripheral to two buffers in memory.

Table 435. 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 B 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 two memory buffers.

22.5.5 Interleaved transfers

One use for the SRCINC and DSTINC configurations (located in the channel transfer configuration registers, XFERCFGn) is to handle data in a buffer such that it is interleaved with other data.

For example, if four data samples from several peripherals should be interleaved into a single data structure, it may be done while the data is being read in by the DMA. Setting SRCINC to 4x width for each channel involved will allow room for four samples in a row in the buffer memory. The DMA will place data for each successive value at the next location for that peripheral.

The reverse of this process could be done using DSTINC to de-interleave combined data from the buffer and send it to several peripherals or locations.

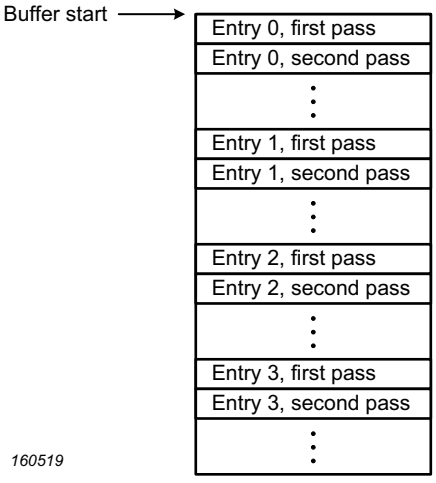


Fig 70. Interleaved transfer in a single buffer

22.5.6 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 can start out like the example for a Ping-Pong transfer Table 435. The difference can be that descriptor B will not link back to descriptor A, but will continue on to another different descriptor. It can continue as long as wanted, and can be ended anywhere, or linked back to any point to repeat a sequence of descriptors. But, any descriptor not currently in use can be altered by software as well.

22.5.7 Address alignment for data transfer

Transfers of 16-bit width requires 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.

22.5.8 Channel chaining

Channel chaining is a feature which allows completion of a DMA transfer on channel x to trigger a DMA transfer on channel y. This feature, for example can be used to have DMA channel x reading n bytes from UART to memory, and then have DMA channel y transferring the received bytes to the CRC engine, without any action required from the ARM core.

To use channel chaining, first configure DMA channels x and y as if no channel chaining would be used.

- For channel x:
  - If channel x is configured to auto reload the descriptor on exhausting of the descriptor (bit RELOAD in the transfer configuration of the descriptor is set), then enable 'clear trigger on descriptor exhausted' by setting bit CLRTRIG in the channel's transfer configuration in the descriptor.
- For channel y:
  - Configure the input trigger input multiplexer register (DMA_ITRIG_INMUX[0:21]) for channel y to use any of the available DMA trigger multiplexers (DMA trigger multiplexer 0/1).
  - Configure the chosen DMA trigger multiplexer to select DMA channel x.
  - Enable hardware triggering by setting bit HWTRIGEN in the channel configuration register.
  - Set the trigger type to edge sensitive by clearing bit TRIGTYPE in the channel configuration register
  - Configure the trigger edge to falling edge by clearing bit TRIGPOL in the channel configuration register

Note: After completion of channel x the descriptor may be reloaded (if configured so), but remains un-triggered. To configure the chain to auto-trigger itself, setup channels x and y for channel chaining as described above. In addition to that

- A Ping-Pong configuration for both channel x and y is recommended, so that data currently moved by channel y is not altered by channel x.
- For channel x:
  - Configure the input trigger input multiplexer register (DMA_ITRIG_INMUX[0:21]) for channel y to use the same DMA trigger multiplexer as chosen for channel y.

- Enable hardware triggering by setting bit HWTRIGEN in the channel configuration register.
- Set the trigger type to edge sensitive by clearing bit TRIGTYPE in the channel configuration register.
- Configure the trigger edge to falling edge by clearing bit TRIGPOL in the channel configuration register.

22.5.8.1 DMA in reduced power mode

DMA in sleep mode

In sleep mode, the DMA can operate and access all enabled SRAM blocks, without waking up the CPU.

DMA in deep-sleep mode

Some peripherals support DMA service during deep-sleep mode without waking up the CPU or the rest of the device. These peripherals are the Flexcomm Interface functions that include FIFO support (USART, SPI, and I2S).

These wake-ups are based on peripheral FIFO levels, not directly related to peripheral DMA requests and interrupts. See Section 8.5.98 for more information

22.6 Register description

The DMA registers are grouped into DMA control, interrupt and status registers and DMA channel registers. DMA transfers are controlled by a set of three registers per channel, the CFG[0:29], CTRLSTAT[0:29], and XFERCFG[0:29] registers. 2 DMA controllers are present: DMA0 and DMA1.

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

Table 436. Register overview: 2 DMA controllers: DMA0 controller (base address = 0x4008 2000) + DMA1 controller (base address = 0x400A 7000)

Name	Access	Offset	Description	Reset value	Section
Global control and status registers					
CTRL	R/W	0x000	DMA control.	0	22.6.1
INTSTAT	RO	0x004	Interrupt status.	0	22.6.2
SRAMBASE	R/W	0x008	SRAM address of the channel descriptors table.	0	22.6.3
Shared registers					
ENABLESET0	R/W	0x020	Channel enable read and Set for all DMA channels.	0	22.6.4
ENABLECLR0	WO	0x028	Channel enable clear for all DMA channels	NA	22.6.5
ACTIVE0	RO	0x030	Channel active status for all DMA channels.	0	22.6.6
BUSY0	RO	0x038	Channel busy status for all DMA channels.	0	22.6.7
ERRINT0	R/W	0x040	Error interrupt status for all DMA channels.	0	22.6.8
INTENSET0	R/W	0x048	Interrupt enable read and Set for all DMA channels.	0	22.6.9
INTENCLR0	WO	0x050	Interrupt enable clear for all DMA channels.	NA	22.6.10
INTA0	R/W	0x058	Interrupt A status for all DMA channels.	0	22.6.11

Table 436. Register overview: 2 DMA controllers: DMA0 controller (base address = 0x4008 2000) + DMA1 controller (base address = 0x400A 7000) *…continued*

Name	Access	Offset	Description	Reset value	Section
INTB0	R/W	0x060	Interrupt B status for all DMA channels.	0	22.6.12
SETVALID0	WO	0x068	Set ValidPending control bits for all DMA channels.	NA	22.6.13
SETTRIG0	WO	0x070	Set trigger control bits for all DMA channels.	NA	22.6.14
ABORT0	WO	0x078	Channel abort control for all DMA channels.	NA	22.6.15
Channel 0 registers					
CFG0	R/W	0x400	Configuration register for DMA channel 0.	0	22.6.16
CTLSTAT0	RO	0x404	Control and status register for DMA channel 0.	0	22.6.17
XFERCFG0	R/W	0x408	Transfer configuration register for DMA channel 0.	0	22.6.18
Channel 1 registers					
CFG1	R/W	0x410	Configuration register for DMA channel 1.	0	22.6.16
CTLSTAT1	RO	0x414	Control and status register for DMA channel 1.	0	22.6.17
XFERCFG1	R/W	0x418	Transfer configuration register for DMA channel 1.	0	22.6.18
Channel 2 registers					
CFG2	R/W	0x420	Configuration register for DMA channel 2.	0	22.6.16
CTLSTAT2	RO	0x424	Control and status register for DMA channel 2.	0	22.6.17
XFERCFG2	R/W	0x428	Transfer configuration register for DMA channel 2.	0	22.6.18
Channel 3 registers					
CFG3	R/W	0x430	Configuration register for DMA channel 3.	0	22.6.16
CTLSTAT3	RO	0x434	Control and status register for DMA channel 3.	0	22.6.17
XFERCFG3	R/W	0x438	Transfer configuration register for DMA channel 3.	0	22.6.18
Channel 4 registers					
CFG4	R/W	0x440	Configuration register for DMA channel 4.	0	22.6.16
CTLSTAT4	RO	0x444	Control and status register for DMA channel 4.	0	22.6.17
XFERCFG4	R/W	0x448	Transfer configuration register for DMA channel 4.	0	22.6.18
Channel 5 registers					
CFG5	R/W	0x450	Configuration register for DMA channel 5.	0	22.6.16
CTLSTAT5	RO	0x454	Control and status register for DMA channel 5.	0	22.6.17
XFERCFG5	R/W	0x458	Transfer configuration register for DMA channel 5.	0	22.6.18
Channel 6 registers					
CFG6	R/W	0x460	Configuration register for DMA channel 6.	0	22.6.16
CTLSTAT6	RO	0x464	Control and status register for DMA channel 6.	0	22.6.17
XFERCFG6	R/W	0x468	Transfer configuration register for DMA channel 6.	0	22.6.18
Channel 7 registers					
CFG7	R/W	0x470	Configuration register for DMA channel 7.	0	22.6.16
CTLSTAT7	RO	0x474	Control and status register for DMA channel 7.	0	22.6.17
XFERCFG7	R/W	0x478	Transfer configuration register for DMA channel 7.	0	22.6.18
Channel 8 registers					
CFG8	R/W	0x480	Configuration register for DMA channel 8.	0	22.6.16
CTLSTAT8	RO	0x484	Control and status register for DMA channel 8.	0	22.6.17
XFERCFG8	R/W	0x488	Transfer configuration register for DMA channel 8.	0	22.6.18

Table 436. Register overview: 2 DMA controllers: DMA0 controller (base address = 0x4008 2000) + DMA1 controller (base address = 0x400A 7000) *…continued*

Name	Access	Offset	Description	Reset value	Section
Channel 9 registers					
CFG9	R/W	0x490	Configuration register for DMA channel 9.	0	22.6.16
CTLSTAT9	RO	0x494	Control and status register for DMA channel 9.	0	22.6.17
XFERCFG9	R/W	0x498	Transfer configuration register for DMA channel 9.	0	22.6.18
Channel 10 registers					
CFG10	R/W	0x4A0	Configuration register for DMA channel 10.	0	22.6.16
CTLSTAT10	RO	0x4A4	Control and status register for DMA channel 10.	0	22.6.17
XFERCFG10	R/W	0x4A8	Transfer configuration register for DMA channel 10.	0	22.6.18
Channel 11registers					
CFG11	R/W	0x4B0	Configuration register for DMA channel 11.	0	22.6.16
CTLSTAT11	RO	0x4B4	Control and status register for DMA channel 11.	0	22.6.17
XFERCFG11	R/W	0x4B8	Transfer configuration register for DMA channel 11.	0	22.6.18
Channel 12 registers					
CFG12	R/W	0x4C0	Configuration register for DMA channel 12.	0	22.6.16
CTLSTAT12	RO	0x4C4	Control and status register for DMA channel 12.	0	22.6.17
XFERCFG12	R/W	0x4C8	Transfer configuration register for DMA channel 12.	0	22.6.18
Channel 13 registers					
CFG13	R/W	0x4D0	Configuration register for DMA channel 13.	0	22.6.16
CTLSTAT13	RO	0x4D4	Control and status register for DMA channel 13.	0	22.6.17
XFERCFG13	R/W	0x4D8	Transfer configuration register for DMA channel 13.	0	22.6.18
Channel 14 registers					
CFG14	R/W	0x4E0	Configuration register for DMA channel 14.	0	22.6.16
CTLSTAT14	RO	0x4E4	Control and status register for DMA channel 14.	0	22.6.17
XFERCFG14	R/W	0x4E8	Transfer configuration register for DMA channel 14.	0	22.6.18
Channel 15 registers					
CFG15	R/W	0x4F0	Configuration register for DMA channel 15.	0	22.6.16
CTLSTAT15	RO	0x4F4	Control and status register for DMA channel 15.	0	22.6.17
XFERCFG15	R/W	0x4F8	Transfer configuration register for DMA channel 15.	0	22.6.18
Channel 16 registers					
CFG16	R/W	0x500	Configuration register for DMA channel 16.	0	22.6.16
CTLSTAT16	RO	0x504	Control and status register for DMA channel 16.	0	22.6.17
XFERCFG16	R/W	0x508	Transfer configuration register for DMA channel 16.	0	22.6.18
Channel 17 registers					
CFG17	R/W	0x510	Configuration register for DMA channel 17.	0	22.6.16
CTLSTAT17	RO	0x514	Control and status register for DMA channel 17.	0	22.6.17
XFERCFG17	R/W	0x518	Transfer configuration register for DMA channel 17.	0	22.6.18
Channel 18 registers					
CFG18	R/W	0x520	Configuration register for DMA channel 18.	0	22.6.16
CTLSTAT18	RO	0x524	Control and status register for DMA channel 18.	0	22.6.17
XFERCFG18	R/W	0x528	Transfer configuration register for DMA channel 18	0	22.6.18

Table 436. Register overview: 2 DMA controllers: DMA0 controller (base address = 0x4008 2000) + DMA1 controller (base address = 0x400A 7000) *…continued*

Name	Access	Offset	Description	Reset value	Section
Channel 19 registers					
CFG19	R/W	0x530	Configuration register for DMA channel 19.	0	22.6.16
CTLSTAT19	RO	0x534	Control and status register for DMA channel 19.	0	22.6.17
XFERCFG19	R/W	0x538	Transfer configuration register for DMA channel 19.	0	22.6.18
Channel 20 registers			Reserved.		
Channel 21 registers					
CFG21	R/W	0x550	Configuration register for DMA channel 21.	0	22.6.16
CTLSTAT21	RO	0x554	Control and status register for DMA channel 21.	0	22.6.17
XFERCFG21	R/W	0x558	Transfer configuration register for DMA channel 21.	0	22.6.18
Channel 22 registers					
CFG22	R/W	0x560	Configuration register for DMA channel 22.	0	22.6.16
CTLSTAT22	RO	0x564	Control and status register for DMA channel 22.	0	22.6.17
XFERCFG22	R/W	0x568	Transfer configuration register for DMA channel 22.	0	22.6.18

22.6.1 Control register

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

Table 437. Control register (CTRL, offset 0x000)

Bit	Symbol	Value	Description	Reset value
0	ENABLE		DMA controller master enable.	0
		0	Disabled. The DMA controller is disabled. It 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

22.6.2 Interrupt status register

The read-only INTSTAT register provides an overview of DMA status. It 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 438. Interrupt status register (INTSTAT, offset 0x004)

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

Table 438. Interrupt status register (INTSTAT, offset 0x004) *…continued*

Bit	Symbol	Value	Description	Reset value
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

22.6.3 SRAM base address register

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

Table 439. SRAM base address register (SRAMBASE, offset 0x008)

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

Each DMA channel has an entry for the channel descriptor in the SRAM table. The values for each channel start at the address offsets found in Table 440. Only the descriptors for channels defined at extraction are used. The contents of each channel descriptor are described in Table 432

Table 440. Channel descriptor map [1]

Offset	Description
0x000	Channel descriptor for DMA channel 0.
0x010	Channel descriptor for DMA channel 1.
0x020	Channel descriptor for DMA channel 2.
0x030	Channel descriptor for DMA channel 3.
0x040	Channel descriptor for DMA channel 4.
0x050	Channel descriptor for DMA channel 5.
0x060	Channel descriptor for DMA channel 6.
0x070	Channel descriptor for DMA channel 7.
0x080	Channel descriptor for DMA channel 8.
0x090	Channel descriptor for DMA channel 9.
0x0A0	Channel descriptor for DMA channel 10.
0x0B0	Channel descriptor for DMA channel 11.
0x0C0	Channel descriptor for DMA channel 12.
0x0D0	Channel descriptor for DMA channel 13.
0x0E0	Channel descriptor for DMA channel 14.
0x0F0	Channel descriptor for DMA channel 15.
0x100	Channel descriptor for DMA channel 16.
0x110	Channel descriptor for DMA channel 17.
0x120	Channel descriptor for DMA channel 18.
0x130	Channel descriptor for DMA channel 19.

Table 440. Channel descriptor map [1] *...continued*

Offset	Description
0x140	Channel descriptor for DMA channel 20.
0x150	Channel descriptor for DMA channel 21.
0x160	Channel descriptor for DMA channel 22.
0x170	Channel descriptor for DMA channel 23.
0x180	Channel descriptor for DMA channel 24.
0x190	Channel descriptor for DMA channel 25.
0x1A0	Channel descriptor for DMA channel 26.
0x1B0	Channel descriptor for DMA channel 27.
0x1C0	Channel descriptor for DMA channel 28.
0x1D0	Channel descriptor for DMA channel 20.
0x1E0	Channel descriptor for DMA channel 30.
0x1F0	Channel descriptor for DMA channel 31.

[1] DMA0 applies to channels 0-22 and DMA1 applies to channels 0-9

22.6.4 Enable read and set register 0

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 441. Enable read and set register 0 (ENABLESET0, offset = 0x020))

Bit	Symbol	Description	Reset value
31:0	ENA	Enable for DMA channels. Bit n enables or disables DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = disabled. 1 = enabled.	0

22.6.5 Enable clear register

The ENABLECLR0 register is used to clear the enable of one or more DMA channels. This register is write-only

Table 442. Enable clear register 0 (ENABLECLR0, offset = 0x028)

Bit	Symbol	Description	Reset value
31:0	CLR	Writing ones to this register clears the corresponding bits in ENABLESET0. Bit n clears the channel enable bit n. The number of bits = number of DMA channels in this device. Other bits are reserved.	NA

22.6.6 Active status register

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

A DMA channel is considered active when a DMA operation has been started but not yet fully completed. The Active status will persist from a DMA operation being started, until the pipeline is empty after end of the last descriptor (when there is no reload). An active channel may be aborted by software by setting the appropriate bit in one of the Abort register. See Section 22.6.15 “Abort register”.

Table 443. Active status register 0 (ACTIVE0, offset = 0x030)

Bit	Symbol	Description	Reset value
31:0	ACT	Active flag for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = not active. 1 = active.	0

22.6.7 Busy status register

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

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

Table 444. Busy status register 0 (BUSY0, offset = 0x038)

Bit	Symbol	Description	Reset value
31:0	BSY	Busy flag for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = not busy. 1 = busy.	0

22.6.8 Error interrupt registers

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 445. Error interrupt register 0, (ERRINT0, offset = 0x40)

Bit	Symbol	Description	Reset value
31:0	ERR	Error Interrupt flag for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = error interrupt is not active. 1 = error interrupt is active.	0

22.6.9 Interrupt enable read and set register

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

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

Table 446. Interrupt enable read and set register 0, (INTENSET0, offset = 0x048)

Bit	Symbol	Description	Reset value
31:0	INTEN	Interrupt enable read and set for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = interrupt for DMA channel is disabled. 1 = interrupt for DMA channel is enabled.	0

22.6.10 Interrupt enable clear register

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

Table 447. Interrupt enable clear register 0, (INTENCLR0, offset = 0x050)

Bit	Symbol	Description	Reset value
31:0	CLR	Writing ones to this register clears corresponding bits in the INTENSET0. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved.	NA

22.6.11 Interrupt A register

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

Table 448. Interrupt A register 0, (INTA0, offset = 0x058)

Bit	Symbol	Description	Reset value
31:0	IA	Interrupt A status for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = the DMA channel interrupt A is not active. 1 = the DMA channel interrupt A is active.	NA

22.6.12 Interrupt B register

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

Table 449. Interrupt B register 0, (INTB0, offset = 0x060)

Bit	Symbol	Description	Reset value
31:0	IB	Interrupt B status for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = the DMA channel interrupt B is not active. 1 = the DMA channel interrupt B is active.	NA

22.6.13 Set valid register

The SETVALID0 register allows setting the Valid bit in the CTRLSTAT register for one or more DMA channels. See Section 22.6.17 “Channel control and status registers” for a description of the VALID bit. This register is write-only.

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 450. Set valid 0 register (SETVALID0, offset = 0x068)

Bit	Symbol	Description	Reset value
31:0	SV	SETVALID control for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = no effect. 1 = sets the VALIDPENDING control bit for DMA channel n.	NA

22.6.14 Set trigger register

The SETTRIG0 register allows setting the TRIG bit in the CTRLSTAT register for one or more DMA channel. See Section 22.6.17 “Channel control and status registers” for a description of the TRIG bit, and Section 22.5.1 “DMA requests and triggers” for a general description of triggering. This register is write-only.

Table 451. Set trigger 0 register (SETTRIG0, offset = 0x070)

Bit	Symbol	Description	Reset value
31:0	TRIG	Set Trigger control bit for DMA channel 0. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = no effect. 1 = sets the TRIG bit for DMA channel n.	NA

22.6.15 Abort register

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. It prevents the channel from restarting an incomplete operation when it is enabled again. This register is write-only.

Table 452. Abort 0 register (ABORT0, offset = 0x078)

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

22.6.16 Channel configuration register

The CFGn register contains various configuration options for DMA channel n. See Table 454 for a summary of trigger options.

Table 453. Configuration registers for channel 0 to 22 ((CFG[0:22], offset 0x400 (CFG0) to 0x560 (CFG22))

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

Table 453. Configuration registers for channel 0 to 22 ((CFG[0:22], offset 0x400 (CFG0) to 0x560 (CFG22)) *…continued*

Bit	Symbol	Value	Description	Reset value
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
11:8	BURSTPOWER		Burst Power is used in two ways. It always selects the address wrap size when SRCBURSTWRAP and/or DSTBURSTWRAP modes are selected (see descriptions elsewhere in this register). When the TRIGBURST field elsewhere in this register = 1, Burst Power selects how many transfers are performed for each DMA trigger. This can be used, for example, with peripherals that contain a FIFO that can initiate a DMA operation when the FIFO reaches a certain level. 0000: Burst size = 1 (2^0). 0001: Burst size = 2 (2^1). 0010: Burst size = 4 (2^2). 1010: Burst size = 1024 (2^{10}). This corresponds to the maximum supported transfer count. others: not supported. The total transfer length as defined in the XFERCOUNT bits in the XFERCFG register must be an integer of the burst size. Note that the total number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field).	
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: Source Burst Wrapping disabled. 1: Source Burst Wrapping enabled.	0

Table 453. Configuration registers for channel 0 to 22 ((CFG[0:22], offset 0x400 (CFG0) to 0x560 (CFG22)) *…continued*

Bit	Symbol	Value	Description	Reset value
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: Destination Burst Wrapping disabled. 1: Destination Burst Wrapping enabled.	0
18:16	CHPRIORITY		Priority of this channel when multiple DMA requests are pending. Eight priority levels are supported: 0x0 = highest priority. 0x7 = lowest priority.	0
31:19	-	-	Reserved. Read value is undefined, only zero should be written	NA

Table 454. Trigger setting summary

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

22.6.17 Channel control and status registers

The CTLSTATn register provides status flags specific to DMA channel n.These registers are read-only.

Table 455. Channel control and status registers for channel 0 to 22((CTLSTAT[0:22], offset 0x404 (CTLSTAT0) to offset = 0x564(CTLSTAT22))

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
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

22.6.18 Channel transfer configuration registers

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

See Section 22.5.1.3 “Trigger operational detail”.

Table 456. Channel transfer configuration registers

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	Not set. 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	Set. 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.	

Table 456. Channel transfer configuration registers *…continued*

Bit	Symbol	Value	Description	Reset value
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
9:8	WIDTH		Transfer width used for this DMA channel.	0
		0x0	8-bit. 8-bit transfers are performed (8-bit source reads and destination writes).	
		0x1	16-bit. 6-bit transfers are performed (16-bit source reads and destination writes).	
		0x2	32-bit. 32-bit transfers are performed (32-bit source reads and destination writes).	
		0x3	Reserved. 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.	

Table 456. Channel transfer configuration registers *…continued*

Bit	Symbol	Value	Description	Reset value
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). XFERCOUNT is used to count down during DMA transfer. When one DMA transfer is completed, XFERCOUNT decrements by 1. Example: The total number of DMA transfer to complete is N. The initial value for XFERCOUNT is N-1. XFERCOUNT = N - 1 means there are N transfers to complete. XFEFCOUNT = N - 2 means there are N-1 transfers to complete. ... XFERCOUNT = 1 means there are 2 transfers to complete. XFERCOUNT = 0 means there is 1 transfer to complete. XFERCOUNT = 0x3FF means all transfers are completed. **Remark:** When all transfers are completed, XFERCOUNT value changes from 0 to 0x3FF. The last value 0x3FF does not mean there are 1024 transfers left to complete. If the initial value for XFERCOUNT is 0x3FF (that is, when the XFERCFGn register is programmed), then there are 1024 transfers to complete. The size of each DMA transfer is determined by the WIDTH field of the same XFERCFGn register.	0
31:26	-	-	Reserved. Read value is undefined, only zero should be written.	-

23.1 How to read this chapter

The SDMMC card interface is available on all LPC55S6x/LPC55S2x/LPC552x devices. The SDMMC card interface supports interface to two devices (SD0 and SD1) with the same SDMMC peripheral.

23.2 Features

The SDMMC card interface supports the following features:

- Secure Digital memory protocol commands.
- Secure Digital I/O protocol commands.
- Multimedia Card protocol commands.
- CE-ATA digital protocol commands.
- Command Completion signal and interrupt to processor.
- Completion Signal disable feature.
- Two SD or MMC (4.4), CE-ATA (1.1), or eMMC (4.4) device.
- CRC 2.0 generation and error detection.
- SDIO interrupts in 1-bit and 4-bit modes.
- SDIO suspend and resume operation.
- SDIO read wait.
- Block size of 1 to 65,535 bytes.
- FIFO over-run and under-run prevention by stopping card clock.
- Little-endian mode of AHB operation.
- Internal (bus mastering) DMA.
- Two FIFOs, TX and RX FIFO (FIFO depth = 32 and FIFO data width = 32 bits).

23.3 Basic configuration

The SDMMC interface is configured as follows:

- Clock Section 23.9.2.3 “SDIOCLKCTRL register”:
  - Enable the clock source that will be used, if it is not already running (most oscillators may be turned off when not needed in order to save power).
  - Select the clock source that will be used in the SDIOCLKSEL register. See Section 4.5.58 “SDIO clock divider”.
  - When using phase delay, the input clock selected in the SDIOCLKSEL register must be 2x of the SDIO clock. See Section 23.9.2.3 “SDIOCLKCTRL register” , SDIOCLKCTRL register. For clocking and timing guidelines, see Section 23.9 “Clocking and timing guidelines”.

- Set up the clock divider (SDIOCLKDIV) that follows the clock source selection mux to obtain the desired clock rate. See Section 4.5.58 “SDIO clock divider”.

Remark: The SDIO function clock to the interface can be up to 50 MHZ.

- Enable clock to the peripheral in the AHBCLKCTRL2 register. See Section 4.5.19 “AHB clock control 2”.

- Reset: The peripheral may be specifically reset using the PRESETCTRL1 register, but must be removed from the reset state before continuing. See Section 4.5.8 “Peripheral reset control 1”.
- Pins: Configure pins that will be used for this peripheral in the IOCON register block. See Chapter 15 “LPC55S6x/LPC55S2x/LPC552x I/O Pin Configuration (IOCON)” for IOCON details, and for recommended IOCON settings for the SDIO.
- Interrupts: If interrupts will be used with this peripheral, enable them in the NVIC. See Chapter 3 “LPC55S6x/LPC55S2x/LPC552x Nested Vectored Interrupt Controller (NVIC)”.
- Peripheral internal setup:
  - Configure the SDIO/SDMMC as needed for the application.
- Configure SD0 and/or SD1 in the following registers:
  - PWREN register.
  - CLKENA register.
  - CTYPE when switching between 1-bit and 4-bit modes.
  - When sending commands, bits 20-16 of the CMD register must be programmed with 0 or 1 for SD0 or SD1 respectively.
- The delay values on the sample and drive inputs and outputs can be adjusted using the SDIOCLKCTRL register in the SYSCON block. See Section 4.5.68 “SDIO CCLKIN phase and delay control”.

  Remark: The LPC55S6x/LPC55S2x/LPC552x device cannot simultaneously access two SD cards.

23.4 Pin description

- Bus Interface Unit (BIU) - Provides AHB and DMA interfaces for register and data read/writes.
- Card Interface Unit (CIU) - Handles the card protocols and provides clock management.
- Internal MCI DMA controller: AHB bus mastering DMA controller.

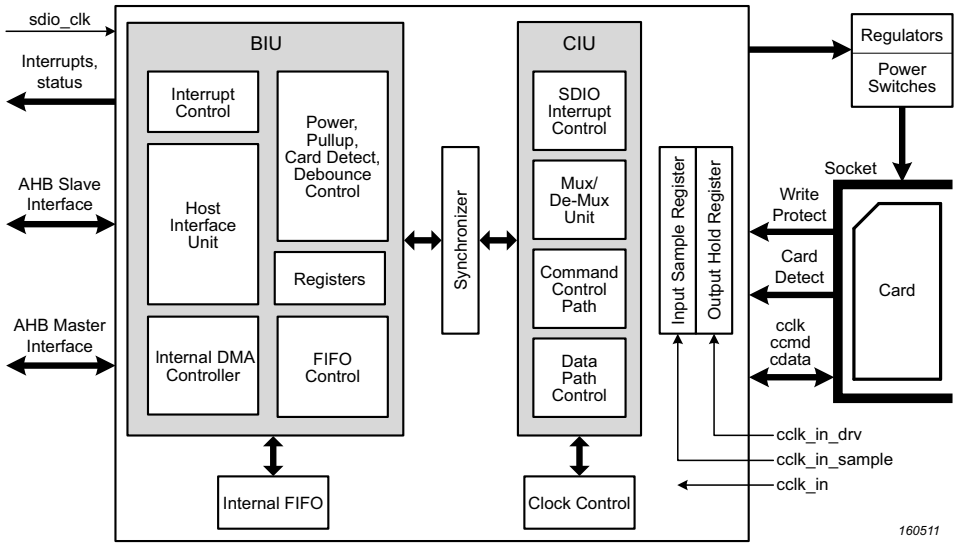


Fig 71. SD/MMC block diagram

23.5 Pin description

Table 457. SD/MMC CARD pin description

Pin function	Type	Description
SD0_CLK, SD1_CLK	O	SD/SDIO/MMC clock.
SD0_CARD_DET_N, SD1_CARD_DET_N	I	SDIO card detect for single slot. A 0 represents the presence of a card.
SD0_WR_PRT	I	SDIO card write protect. A 1 represents write is protected.
SD0_CMD, SD1_CMD	O/I	Command input/output.
SD0_D[7:0], SD1_D[3:0]	O/I	Data input/output for data lines DAT[7:0].
SD0_POW_EN, SD1_POW_EN	O	SD/SDIO/MMC slot power enable.
SD1_BACKEND_PWR	O	Back-end power supply for embedded device. Controls back-end power supply for one embedded device; this bit does not control the VDDH of the host controller. A register bit enables software programming. The value on this register controls switching on and off of power to embedded device.
SD0_CARD_INT_N, SD1_CARD_INT_N	I	Card interrupt line. This pin is used to indicate a card interrupt, which is sampled even when the clock to the card is switched off. Connected to the eSDIO card interrupt line; it is defined only for eSDIO.

Table 458. Suggested pin settings for SD0_CLK, SD0_CMD, SD0_Dn, SD1_CLK, SD1_CMD, SD1_Dn

IOCON bit(s)	Type D pin	Type A pin
10	Not used, set to 0.	Analog switch is open (disabled). Set to 0.
9	Controls open-drain mode. Set to 0.	Same as type D.
8	DIGIMODE: Set to 1.	Same as type D.

Table 458. Suggested pin settings for SD0_CLK, SD0_CMD, SD0_Dn, SD1_CLK, SD1_CMD, SD1_Dn *…continued*

IOCON bit(s)	Type D pin	Type A pin
7	INVERT: Set to 0.	Same as type D.
6	SLEW: Set to 1.	Same as type D.
5:4	MODE: set to 0.	Same as type D.
3:0	FUNC: Must select the correct function for this peripheral.	Same as type D.
General comment	A good choice for SDIO functions.	A potential choice, performance may be reduced by absence of the SLEW function.

23.6 Register description

Figure 71 shows the memory map of the SDIO peripheral.

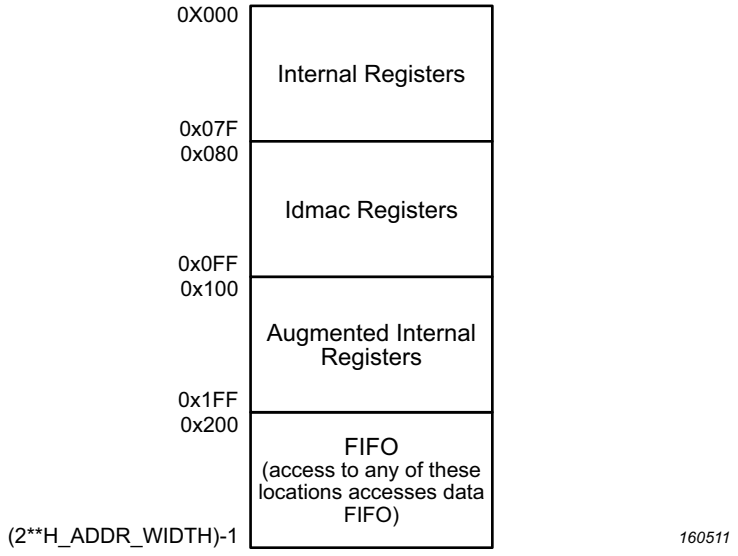


Fig 72. SDIO memory map

Table 459. Register overview: SDMMC (base address: 0x4009 B000)

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	Control.	0	23.6.1
PWREN	R/W	0x004	Power enable.	0	23.6.2
CLKDIV	R/W	0x008	Clock divider.	0	23.6.3
CLKENA	R/W	0x010	Clock enable.	0	23.6.4
TMOUT	R/W	0x014	Time-out.	0xFFFF FF40	23.6.5
CTYPE	R/W	0x018	Card type.	0	23.6.6
BLKSIZ	R/W	0x01C	Block size.	0x200	23.6.7
BYTCNT	R/W	0x020	Byte count.	0x200	23.6.8
INTMASK	R/W	0x024	Interrupt mask.	0	23.6.9
CMDARG	R/W	0x028	Command argument.	0	23.6.10
CMD	R/W	0x02C	Command.	0	23.6.11
RESP0	R	0x030	Response 0.	0	23.6.12
RESP1	R	0x034	Response 1.	0	23.6.13
RESP2	R	0x038	Response 2.	0	23.6.14
RESP3	R	0x03C	Response 3.	0	23.6.15
MINTSTS	R	0x040	Masked interrupt status.	0	23.6.16
RINTSTS	R/W	0x044	Raw interrupt status.	0	23.6.17
STATUS	R	0x048	Status.	0x406	23.6.18
FIFOTH	R/W	0x04C	FIFO threshold watermark.	0x0F80 0000	23.6.19
CDETECT	R	0x050	Card detect.	0	23.6.20

Table 459. Register overview: SDMMC (base address: 0x4009 B000) *…continued*

Name	Access	Offset	Description	Reset value	Section
WRTPRT	R	0x054	Write protect.	0	23.6.21
TCBCNT	R	0x05C	Transferred CIU card byte count.	0	23.6.22
TBBCNT	R	0x060	Transferred host to BIU-FIFO byte count.	0	23.6.23
DEBNCE	R/W	0x064	De-bounce count.	0xFFFFFF	23.6.24
RST_N	R/W	0x078	Hardware reset.	0x1	23.6.25
BMOD	R/W	0x080	Bus mode.	0	23.6.26
PLDMND	W	0x084	Poll demand.	0	23.6.27
DBADDR	R/W	0x088	Descriptor list base address.	0	23.6.28
IDSTS	R/W	0x08C	Internal DMAC status.	0	23.6.29
IDINTEN	R/W	0x090	Internal DMAC interrupt enable.	0	23.6.30
DSCADDR	R	0x094	Current host descriptor address.	0	23.6.31
BUFADDR	R	0x098	Current buffer descriptor address.	0	23.6.32
CARDTHRCTL	R/W	0x100	Card threshold control. Controls whether the host controller initiates transfers depending on the FIFO level.	0	23.6.33
BACKENDPWR	R/W	0x104	Power control.	0	23.6.34
FIFOx	R/W	≥ 0x200	Data FIFO read/write; if address is equal or greater than 0x100, then FIFO is selected as long as device is selected. Address 0x100 and above are mapped to the data FIFO. More than one address is mapped to the data FIFO so that the FIFO can be accessed using bursts.	-	-

23.6.1 Control register

Table 460. Control register (CTRL, offset 0x000)

Bit	Symbol	Value	Description	Reset value
0	CONTROLLER_RESET		Controller reset. To reset controller, software should set bit to 1. This bit is auto-cleared after two AHB and two cclk_in clock cycles. This resets – BIU/CIU interface – CIU and state machines – ABORT_READ_DATA, SEND_IRQ_RESPONSE, and READ_WAIT bits of Control register – START_CMD bit of Command register Does not affect any registers or DMA interface, or FIFO. or host interrupts.	0
		0	No change.	
		1	Reset. Reset SD/MMC controller.	
1	FIFO_RESET		Fifo reset. To reset FIFO, software should set bit to 1. This bit is auto-cleared after completion of reset operation. auto-cleared after two AHB clocks.	0
		0	No change.	
		1	Reset. Reset to data FIFO To reset FIFO pointers	

Table 460. Control register (CTRL, offset 0x000) *…continued*

Bit	Symbol	Value	Description	Reset value
2	DMA_RESET		DMA reset. To reset DMA interface, software should set bit to 1. This bit is auto-cleared after two AHB clocks.	0
		0	No change.	
		1	Reset. Reset internal DMA interface control logic.	
3	-		Reserved	
4	INT_ENABLE		Global interrupt enable/disable bit. The int port is 1 only when this bit is 1 and one or more unmasked interrupts are set.	0
		0	Disable interrupts.	
		1	Enable interrupts.	
5			Reserved. Always write this bit as 0.	
6	READ_WAIT		Read/wait. For sending read-wait to SDIO cards.	0
		0	Clear read wait.	
		1	Assert read wait.	
7	SEND_IRQ_RESPONSE		Send irq response. This bit automatically clears once response is sent. To wait for MMC card interrupts, the host issues CMD40, and the SD/MMC controller waits for an interrupt response from the MMC card. In the meantime, if the host wants the SD/MMC interface to exit waiting for interrupt state, it can set this bit, at which time the SD/MMC interface command state-machine sends a CMD40 response on the bus and returns to idle state.	0
		0	No change.	
		1	Send auto IRQ response.	
8	ABORT_READ_DATA		Abort read data. Used in SDIO card suspend sequence.	0
		0	No change.	
		1	Abort. After suspend command is issued during read-transfer, software polls card to find when suspend happened. Once suspend occurs, software sets bit to reset data state-machine, which is waiting for next block of data. This bit automatically clears once data state machine resets to idle. Used in SDIO card suspend sequence.	

Table 460. Control register (CTRL, offset 0x000) *…continued*

Bit	Symbol	Value	Description	Reset value
9	SEND_CCSD		Send ccsd. When set, the SD/MMC controller sends CCSD to the CE-ATA device. Software sets this bit only if current command is expecting CCS (that is, RW_BLK) and interrupts are enabled in CE-ATA device. Once the CCSD pattern is sent to device, the SD/MMC interface automatically clears SEND_CCSD bit. It also sets Command Done (CD) bit in RINTSTS register and generates interrupt to host if Command Done interrupt is not masked. NOTE: Once SEND_CCSD bit is set, it takes two card clock cycles to drive the CCSD on the CMD line. Due to this, during the boundary conditions it may happen that CCSD is sent to the CE-ATA device, even if the device signalled CCS.	0
		0	Clear bit if the SD/MMC controller does not reset the bit.	
		1	Send Command Completion Signal Disable (CCSD) to CE-ATA device.	
10	SEND_AUTO_STOP_CCSD		Send auto stop ccsd. NOTE: Always set SEND_AUTO_STOP_CCSD and SEND_CCSD bits together; SEND_AUTO_STOP_CCSD should not be set independent of SEND_CCSD. When set, the SD/MMC interface automatically sends internally generated STOP command (CMD12) to CE-ATA device. After sending internally-generated STOP command, Auto Command Done (ACD) bit in RINTSTS is set and generates interrupt to host if Auto Command Done interrupt is not masked. After sending the CCSD, the SD/MMC interface automatically clears SEND_AUTO_STOP_CCSD bit.	0
		0	Clear this bit if the SD/MMC controller does not reset the bit.	
		1	Send internally generated STOP after sending CCSD to CE-ATA device.	
11	CEATA_DEVICE_INTERRUPT_STATUS		CEATA device interrupt status. Software should appropriately write to this bit after power-on reset or any other reset to CE-ATA device. After reset, usually CE-ATA device interrupt is disabled (nIEN = 1). If the host enables CE-ATA device interrupt, then software should set this bit.	0
		0	Disabled. Interrupts not enabled in CE-ATA device (nIEN = 1 in ATA control register)	
		1	Enabled. Interrupts are enabled in CE-ATA device (nIEN = 0 in ATA control register)	
15:12	-		Reserved	-
16	CARD_VOLTAGE_A0		Controls the state of the SD_VOLT0 pin. SD/MMC card voltage control is not implemented.	0
17	CARD_VOLTAGE_A1		Controls the state of the SD_VOLT1 pin. SD/MMC card voltage control is not implemented.	0
18	CARD_VOLTAGE_A2		Controls the state of the SD_VOLT2 pin. SD/MMC card voltage control is not implemented.	0
24:19	-		Reserved	-

Table 460. Control register (CTRL, offset 0x000) *…continued*

Bit	Symbol	Value	Description	Reset value
25	USE_INTERNAL_DMAC		SD/MMC DMA use.	0
		0	Host. The host performs data transfers through the slave interface	
		1	DMA. Internal DMA used for data transfer	
31:26			Reserved	-

23.6.2 Power enable register

Table 461. Power enable register (PWREN, offset = 0x004)

Bit	Symbol	Description	Reset value
0	POWER_ENABLE[0]	Power on/off switch for card 0; once power is turned on, software should wait for regulator/switch ramp-up time before trying to initialize card. 0 - power off 1 - power on Optional feature: port can be used as general-purpose output on the SD_POW pin.	0
1	POWER_ENABLE[1]	Power on/off switch for card 1; once power is turned on, software should wait for regulator/switch ramp-up time before trying to initialize card. 0 - power off 1 - power on Optional feature: port can be used as general-purpose output on the SD_POW pin.	0
31:2	-	Reserved	-

23.6.3 Clock divider register

Table 462. Clock divider register (CLKDIV, offset = 0x008)

Bit	Symbol	Description	Reset value
7:0	CLK_DIVIDER0	Clock divider-0 value. Clock division is 2*n. For example, value of 0 means divide by 2 * 0 = 0 (no division, bypass), value of 1 means divide by 2 * 1 = 2, value of “FF” means divide by 2 * 255 = 510, and so on.	0
31:8		Reserved	-

23.6.4 Clock enable register

Table 463. Clock enable register (CLKENA, offset = 0x010)

Bit	Symbol	Description	Reset value
0	CCLK0_ENABLE	Clock-enable control for SD card 0 clock. One MMC card clock supported. 0 - Clock disabled 1 - Clock enabled	0
1	CCLK1_ENABLE	Clock-enable control for SD card 1 clock. One MMC card clock supported. 0 - Clock disabled 1 - Clock enabled	0
15:2	-	Reserved	-
16	CCLK0_LOW_POWER	Low-power control for SD card clock. One MMC card clock supported. 0 - Non-low-power mode 1 - Low-power mode; stop clock when card in IDLE (should be normally set to only MMC and SD memory cards; for SDIO cards, if interrupts must be detected, clock should not be stopped).	0
17	CCLK1_LOW_POWER	Low-power control for SD card clock. One MMC card clock supported. 0 - Non-low-power mode 1 - Low-power mode; stop clock when card in IDLE (should be normally set to only MMC and SD memory cards; for SDIO cards, if interrupts must be detected, clock should not be stopped).	0
31:18	-	Reserved	-

23.6.5 Time-out register

Table 464. Time-out register (TMOUT, offset = 0x014)

Bit	Symbol	Description	Reset value
7:0	RESPONSE_TIMEOUT	Response time-out value. Value is in number of card output clocks - cclk_out.	0x40
31:8	DATA_TIMEOUT	Value for card Data Read time-out; same value also used for Data Starvation by Host time-out. Value is in number of card output clocks - cclk_out of selected card. Starvation by Host time-out. Value is in number of card output clocks - cclk_out of selected card.	0xFFFFFF

23.6.6 Card type register

Table 465. Card type register (CTYPE, offset 0x018)

Bit	Symbol	Description	Reset value
0	CARD0_WIDTH0	Indicates if card 0 is 1-bit or 4-bit: 0 - 1-bit mode 1 - 4-bit mode 1 and 4-bit modes only work when 8-bit mode in CARD0_WIDTH1 is not enabled (bit 16 in this register is set to 0).	0
1	CARD1_WIDTH0	Indicates if card 1 is 1-bit or 4-bit: 0 - 1-bit mode 1 - 4-bit mode 1 and 4-bit modes only work when 8-bit mode in CARD1_WIDTH1 is not enabled (bit 16 in this register is set to 0).	0
15:2	-	Reserved	-
16	CARD0_WIDTH1	Indicates if card 0 is 8-bit: 0 - Non 8-bit mode 1 - 8-bit mode.	0
17	CARD1_WIDTH1	Indicates if card 1 is 8-bit: 0 - Non 8-bit mode 1 - 8-bit mode.	0
31:18	-	Reserved	-

23.6.7 Block size register

Table 466. Block size register (BLKSIZ, offset = 0x01C)

Bit	Symbol	Description	Reset value
15:0	BLOCK_SIZE	Block size	0x200
31:16	-	Reserved	-

23.6.8 Byte count register

Table 467. Byte count register (BYTCNT, offset = 0x020)

Bit	Symbol	Description	Reset value
31:0	BYTE_COUNT	Number of bytes to be transferred; should be integer multiple of Block Size for block transfers. For undefined number of byte transfers, byte count should be set to 0. When byte count is set to 0, it is responsibility of host to explicitly send stop/abort command to terminate data transfer.	0x200

23.6.9 Interrupt mask register

Table 468. Interrupt mask register (INTMASK, offset = 0x024)

Bit	Symbol	Description	Reset value
0	CDET	Card detect. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
1	RE	Response error. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
2	CDONE	Command done. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
3	DTO	Data transfer over. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
4	TXDR	Transmit FIFO data request. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
5	RXDR	Receive FIFO data request. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
6	RCRC	Response CRC error. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
7	DCRC	Data CRC error. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
8	RTO	Response time-out. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
9	DRTO	Data read time-out. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
10	HTO	Data starvation-by-host time-out (HTO). Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
11	FRUN	FIFO underrun/overrun error. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
12	HLE	Hardware locked write error. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
13	SBE	Start-bit error. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
14	ACD	Auto command done. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
15	EBE	End-bit error (read)/Write no CRC. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
16	SDIO_INT_MASK	Mask SDIO interrupt. When masked, SDIO interrupt detection for card is disabled. A 0 masks an interrupt, and 1 enables an interrupt. In MMC-Ver3.3-only mode, this bit is always 0.	0
31:17		Reserved	

23.6.10 Command argument register

Table 469. Command argument register (CMDARG, offset = 0x028)

Bit	Symbol	Description	Reset value
31:0	CMD_ARG	Value indicates command argument to be passed to card.	0

23.6.11 Command register

Table 470. Command register (CMD, offset 0x02C)

Bit	Symbol	Value	Description	Reset value
5:0	CMD_INDEX		Command index	0
6	RESPONSE_EXPECT		Response expect	0
		0	None. No response expected from card	
		1	Expected. Response expected from card	
7	RESPONSE_ LENGTH		Response length	0
		0	Short. Short response expected from card	
		1	Long. Long response expected from card	
8	CHECK_RESPONSE_CRC		Check response CRC. Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller.	0
		0	Do not check response CRC	
		1	Check response CRC	
9	DATA_EXPECTED		Data expected	0
		0	None. No data transfer expected (read/write)	
		1	Data. Data transfer expected (read/write)	
10	READ_WRITE		Read/write. Don't care if no data expected from card.	0
		0	Read from card	
		1	Write to card	
11	TRANSFER_MODE		Transfer mode. Don't care if no data expected.	0
		0	Block data transfer command	
		1	Stream data transfer command	
12	SEND_AUTO_STOP		Send auto stop. When set, the SD/MMC interface sends stop command to SD_MMC_CEATA cards at end of data transfer. Refer to Table 493 to determine: - when SEND_AUTO_STOP bit should be set, since some data transfers do not need explicit stop commands - open-ended transfers that software should explicitly send to stop command Additionally, when “resume” is sent to resume - suspended memory access of SD-Combo card - bit should be set correctly if suspended data transfer needs SEND_AUTO_STOP. Don't care if no data expected from card.	0
		0	No stop command sent at end of data transfer	
		1	Send stop command at end of data transfer	
13	WAIT_PRVDATA_COMPLETE		Wait prvdata complete. The WAIT_PRVDATA_COMPLETE = 0 option typically used to query status of card during data transfer or to stop current data transfer.	0
		0	Send. Send command at once, even if previous data transfer has not completed.	
		1	Wait. Wait for previous data transfer completion before sending command.	

Table 470. Command register (CMD, offset 0x02C) *…continued*

Bit	Symbol	Value	Description	Reset value
14	STOP_ABORT_CMD		Stop abort command. When open-ended or predefined data transfer is in progress, and host issues stop or abort command to stop data transfer, bit should be set so that command/data state-machines of CIU can return correctly to idle state. This is also applicable for Boot mode transfers. To Abort boot mode, this bit should be set along with CMD[26] = DISABLE_BOOT.	0
		0	Disabled. Neither stop nor abort command to stop current data transfer in progress. If abort is sent to function-number currently selected or not in data-transfer mode, then bit should be set to 0	
		1	Enabled. Stop or abort command intended to stop current data transfer in progress.	
15	SEND_INITIALIZATION		Send initialization. After power on, 80 clocks must be sent to card for initialization before sending any commands to card. Bit should be set while sending first command to card so that controller will initialize clocks before sending command to card. This bit should not be set for either of the boot modes (alternate or mandatory).	0
		0	No. Do not send initialization sequence (80 clocks of 1) before sending this command.	
		1	Send. Send initialization sequence before sending this command.	
20:16	CARD_NUMBER	0 or 1	Specifies the card number of SDCARD for which the current Command is being executed	0
21	UPDATE_CLOCK_REGISTERS_ONLY		Update clock registers only. Following register values transferred into card clock domain: CLKDIV, CLRSRC, CLKENA. Changes card clocks (change frequency, truncate off or on, and set low-frequency mode); provided in order to change clock frequency or stop clock without having to send command to cards. During normal command sequence, when UPDATE_CLOCK_REGISTERS_ONLY = 0, following control registers are transferred from BIU to CIU: CMD, CMDARG, TMOUT, CTYPE, BLKSIZ, BYTCNT. CIU uses new register values for new command sequence to card(s). When bit is set, there are no Command Done interrupts because no command is sent to SD_MMC_CEATA cards.	0
		0	Normal. Normal command sequence	
		1		

Table 470. Command register (CMD, offset 0x02C) *…continued*

Bit	Symbol	Value	Description	Reset value
22	READ_CEATA_DEVICE		Read ceata device. Software should set this bit to indicate that CE-ATA device is being accessed for read transfer. This bit is used to disable read data time-out indication while performing CE-ATA read transfers. Maximum value of I/O transmission delay can be no less than 10 seconds.The SD/MMC interface should not indicate read data time-out while waiting for data from CE-ATA device.	0
		0	No read. Host is not performing read access (RW_REG or RW_BLK) towards CE-ATA device.	
		1	Read. Host is performing read access (RW_REG or RW_BLK) towards CE-ATA device.	
23	CCS_EXPECTED		CCS expected. If the command expects Command Completion Signal (CCS) from the CE-ATA device, the software should set this control bit. The SD/MMC controller sets the Data Transfer Over (DTO) bit in the RINTSTS register and generates an interrupt to the host if the Data Transfer Over interrupt is not masked.	0
		0	Disabled. Interrupts are not enabled in CE-ATA device (nIEN = 1 in ATA control register), or command does not expect CCS from device.	
		1	Enabled. Interrupts are enabled in CE-ATA device (nIEN = 0), and RW_BLK command expects command completion signal from CE-ATA device.	
24	ENABLE_BOOT	-	Enable Boot - this bit should be set only for mandatory boot mode. When Software sets this bit along with START_CMD, CIU starts the boot sequence for the corresponding card by asserting the CMD line low. Do NOT set DISABLE_BOOT and ENABLE_BOOT together.	0
25	EXPECT_BOOT_ACK	-	Expect Boot Acknowledge. When Software sets this bit along with ENABLE_BOOT, CIU expects a boot acknowledge start pattern of 0-1-0 from the selected card.	0
26	DISABLE_BOOT		Disable Boot. When software sets this bit along with START_CMD, CIU terminates the boot operation. Do NOT set DISABLE_BOOT and ENABLE_BOOT together.	0
		0		
		1		
27	BOOT_MODE		Boot Mode.	0
		0	Mandatory boot operation.	
		1	Alternate boot operation.	
28	VOLT_SWITCH		Voltage switch bit.	0
		0	Disabled. No voltage switching.	
		1	Enabled. Voltage switching enabled; must be set for CMD11 only.	

Table 470. Command register (CMD, offset 0x02C) *…continued*

Bit	Symbol	Value	Description	Reset value
29	USE_HOLD_REG		Use Hold register.	0
		0	CMD and DATA sent to card bypassing HOLD register.	
		1	CMD and DATA sent to card through the HOLD register. Hold settings applied through the SDIOCLKCTRL register in Syscon. See Section 4.5.68 “SDIO CCLKIN phase and delay control”.	
30	-		Reserved	-
31	START_CMD	-	Start command. Once command is taken by CIU, this bit is cleared. When bit is set, host should not attempt to write to any command registers. If write is attempted, hardware lock error is set in raw interrupt register. Once command is sent and response is received from SD_MMC_CEATA cards, Command Done bit is set in the raw interrupt register.	0

23.6.12 Response register 0

Table 471. Response register 0 (RESP0, offset 0x030)

Bit	Symbol	Description	Reset value
31:0	RESPONSE0	Bit[31:0] of response	0

23.6.13 Response register 1

Table 472. Response register 1 (RESP1, offset 0x034)

Bit	Symbol	Description	Reset value
31:0	RESPONSE1	Register represents bit[63:32] of long response. When CIU sends auto-stop command, then response is saved in register. Response for previous command sent by host is still preserved in Response 0 register. Additional auto-stop issued only for data transfer commands, and response type is always “short” for them. For information on when CIU sends auto-stop commands, refer to Section 23.7.2 “Auto-Stop”.	0

23.6.14 Response register 2

Table 473. Response register 2 (RESP2, offset 0x038)

Bit	Symbol	Description	Reset value
31:0	RESPONSE2	Bit[95:64] of long response	0

23.6.15 Response register 3

Table 474. Response register 3 (RESP3, offset 0x03C)

Bit	Symbol	Description	Reset value
31:0	RESPONSE3	Bit[127:96] of long response	0

23.6.16 Masked Interrupt Status register

Table 475. Masked Interrupt Status register (MINTSTS, offset 0x040)

Bit	Symbol	Description	Reset value
0	CDET	Card detect. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
1	RE	Response error. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
2	CDONE	Command done. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
3	DTO	Data transfer over. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
4	TXDR	Transmit FIFO data request. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
5	RXDR	Receive FIFO data request. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
6	RCRC	Response CRC error. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
7	DCRC	Data CRC error. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
8	RTO	Response time-out. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
9	DRTO	Data read time-out. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
10	HTO	Data starvation-by-host time-out (HTO). Interrupt enabled if corresponding bit in interrupt mask register is set.	0
11	FRUN	FIFO underrun/overrun error. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
12	HLE	Hardware locked write error. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
13	SBE	Start-bit error. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
14	ACD	Auto command done. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
15	EBE	End-bit error (read)/write no CRC. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
16	SDIO_INTERRUPT	Interrupt from SDIO card. SDIO interrupt for card enabled if corresponding SDIO_INT_MASK bit is set in Interrupt Mask register (INTMASK). Mask bit 1 enables interrupt; 0 masks interrupt. 0 - No SDIO interrupt from card 1 - SDIO interrupt from card In MMC-Ver3.3-only mode, this bit is always 0.	0
31:17	-	Reserved	-

23.6.17 Raw interrupt status register

Table 476. Raw interrupt status register (RINTSTS, offset 0x044)

Bit	Symbol	Description	Reset value
0	CDET	Card detect. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
1	RE	Response error. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
2	CDONE	Command done. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
3	DTO	Data transfer over. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
4	TXDR	Transmit FIFO data request. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
5	RXDR	Receive FIFO data request. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
6	RCRC	Response CRC error. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
7	DCRC	Data CRC error. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
8	RTO_BAR	Response time-out (RTO)/Boot Ack Received (BAR). Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
9	DRTO_BDS	Data read time-out (DRTO)/Boot Data Start (BDS). Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
10	HTO	Data starvation-by-host time-out (HTO). Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
11	FRUN	FIFO underrun/overrun error. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
12	HLE	Hardware locked write error. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
13	SBE	Start-bit error. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
14	ACD	Auto command done. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
15	EBE	End-bit error (read)/write no CRC. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
16	SDIO_INTERRUPT	Interrupt from SDIO card. Writes to these bits clear them. Value of 1 clears bit and 0 leaves bit intact. 0 - No SDIO interrupt from card 1 - SDIO interrupt from card In MMC-Ver3.3-only mode, bits always 0. Bits are logged regardless of interrupt-mask status.	0
31:17		Reserved.	-

23.6.18 Status register

Table 477. Status register (STATUS, offset 0x048)

Bit	Symbol	Description	Reset value
0	FIFO_RX_WATERMARK	FIFO reached Receive watermark level; not qualified with data transfer.	0
1	FIFO_TX_WATERMARK	FIFO reached Transmit watermark level; not qualified with data transfer.	1
2	FIFO_EMPTY	FIFO is empty status.	1
3	FIFO_FULL	FIFO is full status.	0
7:4	CMDFSMSTATES	Command FSM states: 0 - Idle 1 - Send init sequence 2 - Tx cmd start bit 3 - Tx cmd tx bit 4 - Tx cmd index + arg 5 - Tx cmd crc7 6 - Tx cmd end bit 7 - Rx resp start bit 8 - Rx resp IRQ response 9 - Rx resp tx bit 10 - Rx resp cmd idx 11 - Rx resp data 12 - Rx resp crc7 13 - Rx resp end bit 14 - Cmd path wait NCC 15 - Wait; CMD-to-response turnaround NOTE: The command FSM state is represented using 19 bits. The STATUS register(7:4) has 4 bits to represent the command FSM states. Using these 4 bits, only 16 states can be represented. Thus three states cannot be represented in the STATUS(7:4) register. The three states that are not represented in the STATUS register(7:4) are: - Bit 16 - Wait for CCS - Bit 17 - Send CCSD - Bit 18 - Boot Mode Due to this, while command FSM is in “Wait for CCS state” or “Send CCSD” or “Boot Mode”, the STATUS register indicates status as 0 for the bit field 7:4.	0
8	DATA_3_STATUS	Raw selected card_data[3]; checks whether card is present. 0 - card not present 1 - card present	0
9	DATA_BUSY	Inverted version of raw selected card_data[0]. 0 - card data not busy 1 - card data busy	0
10	DATA_STATE_MC_BUSY	Data transmit or receive state-machine is busy.	1
16:11	RESPONSE_INDEX	Index of previous response, including any auto-stop sent by core.	0
29:17	FIFO_COUNT	FIFO count - Number of filled locations in FIFO.	0
30	DMA_ACK	DMA acknowledge signal state.	0
31	DMA_REQ	DMA request signal state.	0

23.6.19 FIFO threshold watermark register

Table 478. FIFO threshold watermark register (FIFOTH, offset 0x04C)

Bit	Symbol	Value	Description	Reset value
11:0	TX_WMARK	-	FIFO threshold watermark level when transmitting data to card. When FIFO data count is less than or equal to this number, DMA/FIFO request is raised. If Interrupt is enabled, then interrupt occurs. During end of packet, request or interrupt is generated, regardless of threshold programming. In non-DMA mode, when transmit FIFO threshold (TXDR) interrupt is enabled, then interrupt is generated instead of DMA request. During end of packet, on last interrupt, host is responsible for filling FIFO with only required remaining bytes (not before FIFO is full or after CIU completes data transfers, because FIFO may not be empty). In DMA mode, at end of packet, if last transfer is less than burst size, DMA controller does single cycles until required bytes are transferred. 12 bits - 1 bit less than FIFO-count of STATUS register, which is 13 bits. Limitation: TX_WMARK >= 1; Recommended value: TX_WMARK = 16; (means less than or equal to FIFO_DEPTH/2).	0
15:12	-		Reserved	-
27:16	RX_WMARK		FIFO threshold watermark level when receiving data to card. When FIFO data count reaches greater than this number, DMA/FIFO request is raised. During end of packet, request is generated regardless of threshold programming in order to complete any remaining data. In non-DMA mode, when receiver FIFO threshold (RXDR) interrupt is enabled, then interrupt is generated instead of DMA request. During end of packet, interrupt is not generated if threshold programming is larger than any remaining data. It is responsibility of host to read remaining bytes on seeing Data Transfer Done interrupt. In DMA mode, at end of packet, even if remaining bytes are less than threshold, DMA request does single transfers to flush out any remaining bytes before Data Transfer Done interrupt is set. 12 bits - 1 bit less than FIFO-count of STATUS register, which is 13 bits. Limitation: RX_WMARK less than FIFO_DEPTH-2 Recommended: RX_WMARK = 15; (means greater than (FIFO_DEPTH/2) - 1) NOTE: In DMA mode during CCS time-out, the DMA does not generate the request at the end of packet, even if remaining bytes are less than threshold. In this case, there will be some data left in the FIFO. It is the responsibility of the application to reset the FIFO after the CCS time-out.	0

Table 478. FIFO threshold watermark register (FIFOTH, offset 0x04C) *...continued*

Bit	Symbol	Value	Description	Reset value
30:28	DMA_MTS		Burst size of multiple transaction; should be programmed same as DW-DMA controller multiple-transaction-size SRC/DEST_MSIZE.The units for transfers is the H_DATA_WIDTH parameter. A single transfer (dw_dma_single assertion in case of Non DW DMA interface) would be signalled based on this value. Value should be sub-multiple of (RX_WMARK + 1) and (32 - TX_WMARK). For example, if FIFO_DEPTH = 16, FDATA_WIDTH = H_DATA_WIDTH Allowed combinations for MSize and TX_WMARK are: MSize = 1, TX_WMARK = 1-15 MSize = 4, TX_WMARK = 8 MSize = 4, TX_WMARK = 4 MSize = 4, TX_WMARK = 12 MSize = 8, TX_WMARK = 8 MSize = 8, TX_WMARK = 4 Allowed combinations for MSize and RX_WMARK are: MSize = 1, RX_WMARK = 0-14 MSize = 4, RX_WMARK = 3 MSize = 4, RX_WMARK = 7 MSize = 4, RX_WMARK = 11 MSize = 8, RX_WMARK = 7 MSize = 8, RX_WMARK = 11 Recommended: MSize = 8, TX_WMARK = 8, RX_WMARK = 7	0
		0x0	1 transfer	
		0x1	4 transfers	
		0x2	8 transfers	
		0x3	16 transfers	
		0x4	32 transfers	
		0x5	64 transfers	
		0x6	128 transfers	
		0x7	256 transfers	
31:11	-	-	Reserved	-

23.6.20 Card detect register

Table 479. Card detect register (CDETECT, offset 0x050)

Bit	Symbol	Description	Reset value
0	CARD_DETECT[0]	Card 0 detect. 0 represents presence of card.	0
1	CARD_DETECT[1]	Card 1 detect. 0 represents presence of card.	0
31:1	-	Reserved	-

23.6.21 Write protect register

Table 480. Write protect register (WRTPRT, offset 0x054)

Bit	Symbol	Description	Reset value
0	WRITE_PROTECT	Write protect. 1 represents write protection.	0
31:1	-	Reserved	-

23.6.22 Transferred CIU card byte count register

Table 481. Transferred CIU card byte count register (TCBCNT, offset 0x05C)

Bit	Symbol	Description	Reset value
31:0	TRANS_CARD_BYTE_COUNT	Number of bytes transferred by CIU unit to card. Register should be read only after data transfer completes; during data transfer, register returns 0.	0

23.6.23 Transferred host to BIU-FIFO byte count register

Table 482. Transferred host to BIU-FIFO byte count register (TBBCNT, offset 0x060)

Bit	Symbol	Description	Reset value
31:0	TRANS_FIFO_BYTE_COUNT	Number of bytes transferred between Host/DMA memory and BIU FIFO.	0

23.6.24 De-bounce count register

Table 483. De-bounce count register (DEBNCE, offset 0x064)

Bit	Symbol	Description	Reset value
23:0	DEBOUNCE_COUNT	Number of host clocks (SD_CLK) used by de-bounce filter logic for card detect; typical de-bounce time is 5-25 ms.	0xFFFFFF
31:24	-	Reserved.	-

23.6.25 Hardware Reset

Table 484. Hardware reset (RST_N, offset 0x078)

Bit	Symbol	Description	Reset value
0	CARD_RESET	Hardware reset. 1 - Active mode 0 - Reset Toggles state on SD_RST pin. This bit causes the card to enter pre-idle state, which requires it to be re-initialized.	1
31:1	-	Reserved.	-

23.6.26 Bus mode register

Table 485. Bus mode register (BMOD, offset 0x080)

Bit	Symbol	Value	Description	Reset value
0	SWR		Software Reset. When set, the DMA Controller resets all its internal registers. SWR is read/write. It is automatically cleared after 1 clock cycle.	0
1	FB		Fixed Burst. Controls whether the AHB Master interface performs fixed burst transfers or not. When set, the AHB will use only SINGLE, INCR4, INCR8 or INCR16 during start of normal burst transfers. When reset, the AHB will use SINGLE and INCR burst transfer operations. FB is read/write.	0
6:2	DSL		Descriptor Skip Length. Specifies the number of HWord/Word/Dword to skip between two unchained descriptors. This is applicable only for dual buffer structure. DSL is read/write	0
7	DE		SD/MMC DMA Enable. When set, the SD/MMC DMA is enabled. DE is read/write.	0

Table 485. Bus mode register (BMOD, offset 0x080) *…continued*

Bit	Symbol	Value	Description	Reset value
10:8	PBL		Programmable Burst Length. These bits indicate the maximum number of beats to be performed in one SD/MMC DMA transaction. The SD/MMC DMA will always attempt to burst as specified in PBL each time it starts a Burst transfer on the host bus. The permissible values are 1, 4, 8, 16, 32, 64, 128 and 256. This value is the mirror of MSIZE of FIFOTH register. In order to change this value, write the required value to FIFOTH register. This is an encode value as follows.Transfer unit is 32 bit. PBL is a read-only value.	0
		0x0	1 transfer.	
		0x1	4 transfers.	
		0x2	8 transfers.	
		0x3	16 transfers.	
		0x4	32 transfers.	
		0x5	64 transfers.	
		0x6	128 transfers.	
		0x7	256 transfers.	
31:11	-	-	Reserved.	-

23.6.27 Poll demand register

Table 486. Poll demand register (PLDMND, offset 0x084)

Bit	Symbol	Description	Reset value
31:0	PD	Poll Demand. If the OWN bit of a descriptor is not set, the FSM goes to the Suspend state. The host needs to write any value into this register for the SD/MMC DMA state machine to resume normal descriptor fetch operation. This is a write only register. PD bit is write-only.	0

23.6.28 Descriptor list base address register

Table 487. Descriptor list base address register (DBADDR, offset 0x088)

Bit	Symbol	Description	Reset value
31:0	SDL	Start of Descriptor List. Contains the base address of the First Descriptor. The LSB bits [1:0] are ignored and taken as all-zero by the SD/MMC DMA internally. Hence these LSB bits are read-only.	0

23.6.29 Internal DMAC status register

Table 488. Internal DMAC status register (IDSTS, offset 0x08C)

Bit	Symbol	Description	Reset value
0	TI	Transmit interrupt. Indicates that data transmission is finished for a descriptor. Writing a 1 clears this bit.	0
1	RI	Receive interrupt. Indicates the completion of data reception for a descriptor. Writing a 1 clears this bit.	0
2	FBE	Fatal bus error interrupt. Indicates that a Bus Error occurred (IDSTS[12:10]). When this bit is set, the DMA disables all its bus accesses. Writing a 1 clears this bit.	0
3	-	Reserved	-
4	DU	Descriptor unavailable interrupt. This bit is set when the descriptor is unavailable due to OWN bit = 0 (DES0[31] =0). Writing a 1 clears this bit.	0

Table 488. Internal DMAC status register (IDSTS, offset 0x08C) *…continued*

Bit	Symbol	Description	Reset value
5	CES	Card error summary. Indicates the status of the transaction to/from the card; also present in RINTSTS. Indicates the logical OR of the following bits: EBE - End Bit Error RTO - Response Time-out/Boot Ack Time-out RCRC - Response CRC SBE - Start Bit Error DRTO - Data Read Time-out/BDS time-out DCRC - Data CRC for Receive RE - Response Error Writing a 1 clears this bit.	0
7:6	-	Reserved.	-
8	NIS	Normal interrupt summary. Logical OR of the following: IDSTS[0] - Transmit Interrupt IDSTS[1] - Receive Interrupt Only unmasked bits affect this bit. This is a sticky bit and must be cleared each time a corresponding bit that causes NIS to be set is cleared. Writing a 1 clears this bit.	0
9	AIS	Abnormal interrupt summary. Logical OR of the following: IDSTS[2] - Fatal Bus Interrupt IDSTS[4] - DU bit Interrupt IDSTS[5] - Card Error Summary Interrupt Only unmasked bits affect this bit. This is a sticky bit and must be cleared each time a corresponding bit that causes AIS to be set is cleared. Writing a 1 clears this bit.	0
12:10	EB	Error bits. Indicates the type of error that caused a Bus Error. Valid only with Fatal Bus Error bit (IDSTS[2]) set. This field does not generate an interrupt. 001 - Host Abort received during transmission 010 - Host Abort received during reception Others: Reserved. EB is read-only.	
16:13	FSM	DMAC state machine present state. 0 - DMA_IDLE 1 - DMA_SUSPEND 2 - DESC_RD 3 - DESC_CHK 4 - DMA_RD_REQ_WAIT 5 - DMA_WR_REQ_WAIT 6 - DMA_RD 7 - DMA_WR 8 - DESC_CLOSE This field is read-only.	
31:17	-	Reserved.	-

23.6.30 Internal DMAC interrupt enable register

Table 489. Internal DMAC interrupt enable register (IDINTEN, offset 0x090)

Bit	Symbol	Description	Reset value
0	TI	Transmit interrupt enable. When set with Normal Interrupt Summary Enable, Transmit Interrupt is enabled. When reset, Transmit Interrupt is disabled.	0
1	RI	Receive interrupt enable. When set with Normal Interrupt Summary Enable, Receive Interrupt is enabled. When reset, Receive Interrupt is disabled.	0
2	FBE	Fatal bus error enable. When set with Abnormal Interrupt Summary Enable, the Fatal Bus Error Interrupt is enabled. When reset, Fatal Bus Error Enable Interrupt is disabled.	0
3	-	Reserved	-
4	DU	Descriptor unavailable interrupt. When set along with Abnormal Interrupt Summary Enable, the DU interrupt is enabled.	0

Table 489. Internal DMAC interrupt enable register (IDINTEN, offset 0x090) *…continued*

Bit	Symbol	Description	Reset value
5	CES	Card error summary interrupt enable. When set, it enables the Card Interrupt summary.	0
7:6	-	Reserved	-
8	NIS	Normal interrupt summary enable. When set, a normal interrupt is enabled. When reset, a normal interrupt is disabled. This bit enables the following bits: IDINTEN[0] - Transmit Interrupt IDINTEN[1] - Receive Interrupt.	0
9	AIS	Abnormal interrupt summary enable. When set, an abnormal interrupt is enabled. This bit enables the following bits: IDINTEN[2] - Fatal Bus Error Interrupt IDINTEN[4] - DU Interrupt IDINTEN[5] - Card Error Summary Interrupt.	0
31:10	-	Reserved	-

23.6.31 Current host descriptor address register

Table 490. Current host descriptor address register (DSCADDR, offset 0x094)

Bit	Symbol	Description	Reset value
31:0	HDA	Host descriptor address pointer. Cleared on reset. Pointer updated by IDMAC during operation. This register points to the start address of the current descriptor read by the SD/MMC DMA.	0

23.6.32 Current buffer descriptor address register

Table 491. Current buffer descriptor address register (BUFADDR, offset 0x098)

Bit	Symbol	Description	Reset value
31:0	HBA	Host buffer address pointer. Cleared on Reset. Pointer updated by SD/MMC DMA during operation. This register points to the current Data Buffer Address being accessed by the SD/MMC DMA.	0

23.6.33 Card threshold control register

Note: This register is applicable when CARDRDTHREN is set to '1'. See Section 23.8.2.16 “Card read threshold”

Table 492. Card Threshold control register (CARDTHRCTL, offset 0x100)

Bit	Symbol	Value	Description	Reset value
0	CARDRDTHREN		Card read threshold enable.	0
		0	Card read threshold disabled.	
		1	Card Read Threshold enabled. Host Controller initiates Read Transfer only if Card Threshold amount of space is available in receive FIFO. For more information, refer Section 23.8.2.16 “Card read threshold”	
1	BSYCLRINTEN		Busy clear interrupt enable. Note: The application can disable this feature if it does not want to wait for a Busy Clear Interrupt. For example, in a multi-card scenario, the application can switch to the other card without waiting for a busy to be completed. In such cases, the application can use the polling method to determine the status of busy. By default this feature is disabled and backward-compatible to the legacy drivers where polling is used.	0
		0	Busy clear interrupt disabled.	
		1	Busy clear interrupt enabled.	

Table 492. Card Threshold control register (CARDTHRCTL, offset 0x100) *…continued*

Bit	Symbol	Value	Description	Reset value
15:2	-	-	Reserved.	-
23:16	CARDTHRESHOLD		Card threshold size. Sets the read and/or write threshold within the 32-entry FIFOs. This field is applicable when CARDRDTHREN is set to '1'.	0
31:24	-		Reserved.	-

23.6.34 Back-end power register

Controls back-end power to the card application. See Section 23.8.2.17 “Back-end power”.

Table 493. Back-end power register (BACK_END_POWER, offset 0x104)

Bit	Symbol	Value	Description	Reset value
0	BACKENDPWR		Back-end Power control for card application.	0
		0	Back-end power off to card application.	
		1	Back-end power supplied to card application.	
31:24	-	-	Reserved.	-

23.7 Functional description

23.7.1 Power/pull-up control and card detection unit

Signal pull-up resistors can be enabled for the SD pins in IOCON by enabling the pull-up for the pads. The approximate pull-up value for a pin is about 50 kOhm. For designs that need to support legacy MMC cards in open-drain mode, an external pull-up controlled

with a general purpose output and FET will be needed for the CMD line.

Slot power can be controlled with the SD_POW pin and the SD_VOLT[2:0] pins. It is recommended that the slot power regulator is enabled and disabled via the SD_POW pin, which can be directly controlled with bit 0 of the Power enable register (PWREN).

Use of the SD_VOLT[2:0] pins is optional and not needed in a design with a single power supply sourcing the card slot.

The card detection signal is de-bounced based on the number of blocks specified in the De-bounce Count register (DEBNCE). When this signal is connected to the card detect pin of the card slot, then CDETECT register's bit 0 state will be filtered by the number of de-bounce cycles specified in DEBNCE. This guarantees that interrupt related to the card detect signal are de-bounced before occurring.

23.7.2 Auto-Stop

The auto-stop command helps to send an exact number of data bytes using a stream read or write for the MMC, and a multiple-block read or write for SD memory transfer for SD cards. The module internally generates a stop command and is loaded in the command path when the SEND_AUTO_STOP bit is set in the Command register (CMD).

The software should set the SEND_AUTO_STOP bit according to details listed in the table below:

Table 494. SEND_AUTO_STOP bit

Card Type	Transfer Type	Byte Count	SEND_AUTO_STOP bit set	Comments
MMC	Stream read	0	No	Open-ended stream
MMC	Stream read	>0	Yes	Auto-stop after all bytes transfer
MMC	Stream read	0	No	Open-ended stream
MMC	Stream read	>0	Yes	Auto-stop after all bytes transfer
MMC	Single-block read	>0	No	Byte count = 0 is illegal
MMC	Single-block write	>0	No	Byte count = 0 is illegal
MMC	Multiple-block read	0	No	Open-ended multiple block
MMC	Multiple-block read	>0	Yes	Pre-defined multiple block
MMC	Multiple-block write	0	No	Open-ended multiple block
MMC	Multiple-block write	>0	Yes	Pre-defined multiple block
SDMEM	Single-block read	>0	No	Byte count = 0 is illegal
SDMEM	Single-block write	>0	No	Byte count = 0 is illegal
SDMEM	Multiple-block read	0	No	Open-ended multiple block
SDMEM	Multiple-block read	>0	Yes	Auto-stop after all bytes transfer
SDMEM	Multiple-block write	0	No	Open-ended multiple block
SDMEM	Multiple-block write	>0	Yes	Auto-stop after all bytes transfer
SDIO	Single-block read	>0	No	Byte count = 0 is illegal
SDIO	Single-block write	>0	No	Byte count = 0 is illegal
SDIO	Multiple-block read	0	No	Open-ended multiple block
SDIO	Multiple-block read	>0	No	Pre-defined multiple block
SDIO	Multiple-block write	>0	No	Open-ended multiple block
SDIO	Multiple-block write	>0	No	Pre-defined multiple block

The following list conditions for the auto-stop command:

- Stream read for MMC card with byte count greater than 0 - The Module generates an internal stop command and loads it into the command path so that the end bit of the stop command is sent out when the last byte of data is read from the card and no extra data byte is received. If the byte count is less than 6 (48 bits), a few extra data bytes are received from the card before the end bit of the stop command is sent.
- Stream write for MMC card with byte count greater than 0 - The Module generates an internal stop command and loads it into the command path so that the end bit of the stop command is sent when the last byte of data is transmitted on the card bus and no extra data byte is transmitted. If the byte count is less than 6 (48 bits), the data path transmits the data last in order to meet the above condition.
- Multiple-block read memory for SD card with byte count greater than 0 - If the block size is less than 4 (single-bit data bus), 16 (4-bit data bus), or 32 (8-bit data bus), the auto-stop command is loaded in the command path after all the bytes are read. Otherwise, the top command is loaded in the command path so that the end bit of the stop command is sent after the last data block is received.

- Multiple-block write memory for SD card with byte count greater than 0 - If the block size is less than 3 (single-bit data bus), 12 (4-bit data bus), or 24 (8-bit data bus), the auto-stop command is loaded in the command path after all data blocks are transmitted. Otherwise, the stop command is loaded in the command path so that the end bit of the stop command is sent after the end bit of the CRC status is received.

Precaution for CPU software during auto-stop - Whenever an auto-stop command is issued, the CPU software should not issue a new command to the Module until the auto-stop is sent by the Module and the data transfer is complete. If the CPU issues a new command during a data transfer with the auto-stop in progress, an auto-stop command may be sent after the new command is sent and its response is received; this can delay sending the stop command, which transfers extra data bytes. For a stream write, extra data bytes are erroneous data that can corrupt the card data. If the CPU wants to terminate the data transfer before the data transfer is complete, it can issue a stop or abort command, in which case the Module does not generate an auto-stop command.

23.8 Programming the SD/MMC

23.8.1 Software/hardware restrictions

Only one data transfer command should be issued at one time. For CE-ATA devices, if CE-ATA device interrupts are enabled (nIEN = 0), only one RW_MULTIPLE_BLOCK command (RW_BLK) should be issued; no other commands (including a new RW_BLK) should be issued before the data transfer. Over status is set for the outstanding RW_BLK.

Before issuing a new data transfer command, the software should ensure that the card is not busy due to any previous data transfer command. Before changing the card clock frequency, the software must ensure that there are no data or command transfers in progress.

To avoid glitches in the card clock outputs (cclk_out), the software should use the following steps when changing the card clock frequency:

1. Update the Clock Enable register (CLKENA) to disable all clocks. To ensure completion of any previous command before this update, send a command to the CIU to update the clock registers by setting:
   - START_CMD bit.
   - *update clock registers only* bits
   - *wait_previous data complete* bit

   Wait for the CIU to take the command by polling for 0 on the START_CMD bit.
2. Set the START_CMD bit to update the clock divider and/or clock source registers, and send a command to the CIU in order to update the clock registers; wait for the CIU to take the command.
3. Set START_CMD to update the Clock Enable register (CLKENA) in order to enable the required clocks and send a command to the CIU to update the clock registers; wait for the CIU to take the command.

In non-DMA mode, while reading from a card, the data transfer over (RINTSTS[3]) interrupt occurs as soon as the data transfer from the card is over. There still could be some data left in the FIFO, and the RX_WMARK interrupt may or may not occur,

depending on the remaining bytes in the FIFO. Software should read any remaining bytes upon seeing the Data Transfer Over (DTO) interrupt. In DMA mode while reading from a card, the DTO interrupt occurs only after all the FIFO data is flushed to memory by the DMA Interface unit.

While writing to a card in DMA mode, if an undefined-length transfer is selected by setting the Byte Count register (BYTCNT) to 0, the DMA logic will likely request more data than it will send to the card, since it has no way of knowing at which point the software will stop the transfer. The DMA request stops as soon as the DTO is set by the CIU.

If the software issues a CONTROLLER_RESET command by setting control register (CTRL) bit[0] to 1, all the CIU state machines are reset; the FIFO is not cleared. The DMA sends all remaining bytes to the CPU. In addition to a card-reset, if a FIFO reset is also issued, then:

- Any pending DMA transfer on the bus completes correctly.
- DMA data read is ignored.
- Write data is unknown (x).

Additionally, if DMA_RESET is also issued, any pending DMA transfer is abruptly terminated. The DMA controller channel should also be reset and reprogrammed.

If any of the previous data commands do not properly terminate, then the software should issue the FIFO reset in order to remove any residual data, if any, in the FIFO. After asserting the FIFO reset, you should wait until this bit is cleared.

One data-transfer requirement between the FIFO and CPU is that the number of transfers should be a multiple of the FIFO data width (F_DATA_WIDTH), which is 32. So if you want to write only 15 bytes to an SD/MMC/CE-ATA card (BYTCNT), the CPU should write 16 bytes to the FIFO or program the DMA to do 16-byte transfers, if DMA mode is enabled. The software can still program the Byte Count register (BYTCNT) to only 15, at which point only 15 bytes will be transferred to the card. Similarly, when 15 bytes are read from a card, the CPU should still read all 16 bytes from the FIFO.

It is recommended not to change the FIFO threshold register in the middle of data transfers.

23.8.2 Programming sequence

23.8.2.1 Initialization

Once the power and clocks are stable, reset_n should be asserted (active-low) for at least two clocks of clk or cclk_in, whichever is slower. The reset initializes the registers, ports, FIFO-pointers, DMA interface controls, and state-machines in the design. After power-on reset, the software should do the following:

1. After power on reset, configure the SD/MMC pin functions via IOCON, see Chapter 15 “LPC55S6x/LPC55S2x/LPC552x I/O Pin Configuration (IOCON)”.
2. Set masks for interrupts by clearing appropriate bits in the Interrupt Mask register (INTMASK). Set the global INT_ENABLE bit of the control register (CTRL). It is recommended that you write 0xFFFF_FFFF to the Raw Interrupt register in order to clear any pending interrupts before setting the INT_ENABLE bit.

3. Enumerate card stack. Each card is enumerated according to card type; for details, refer Section 23.8.2.2 “Enumerated card stack”. For enumeration, you should restrict the clock frequency to 400 kHz in accordance with SD_MMC/CE-ATA standards.
4. Changing clock. The cards operate at a maximum of 25 MHz (at maximum of 50 MHz in high-speed mode and speed detection using CMD6).
5. Set other IP parameters, which normally do not need to be changed with every command, with a typical value such as time-out values in cclk_out according to SD_MMC/CE-ATA specifications.

   ResponseTimeOut = 0x40.

   DataTimeOut = highest of one of the following:

   – (10 × ((TAAC × Fop) + (100 × NSAC)).
   – CPU FIFO read/write latency from FIFO empty/full.

     FIFO threshold value in bytes in the FIFOTH register. Typically, the threshold value can be set to half the FIFO depth (= 32/2); that is:

   – RX_WMARK = (FIFO_DEPTH/2) - 1;
   – TX_WMARK = FIFO_DEPTH/2.

23.8.2.2 Enumerated card stack

The card stack does the following:

- Enumerates all connected cards.
- Sets the RCA for the connected cards.
- Reads card-specific information.
- Stores card-specific information locally.

Enumerate_Card_Stack - Enumerates the card connected on the module. The card can be of the type MMC, CE-ATA, SD, or SDIO. All types of SDIO cards are supported; that is, SDIO_IO_ONLY, SDIO_MEM_ONLY, and SDIO_COMBO cards. The enumeration sequence includes the following steps:

1. Check if the card is connected.
2. Clear the bits in the card_type register. Clear the register bit for a 1-bit, 4-bit, or 8-bit bus width.
3. Identify the card type; that is, SD, MMC, or SDIO.
   – Send CMD5 first.If a response is received, then the card is SDIO
   – If not, send ACMD41; if a response is received, then the card is SD.
   – Otherwise, the card is an MMC or CE-ATA.
4. Enumerate the card according to the card type.

   Use a clock source with a frequency = Fod (that is, 400 kHz) and use the following enumeration command sequence:

   – SD card - Send CMD0, ACMD41, CMD2, CMD3.
   – SDHC card - send CMD0, SDCMD8, ACMD41, CMD2, CMD3
   – SDIO - Send CMD5; if the function count is valid, CMD3. For the SDIO memory section, follow the same commands as for the SD card.

- MMC - Send CMD0, CMD1, CMD2, CMD3

5. Identify the MMC/CE-ATA device.
   - Selecting ATA mode for a CE-ATA device.
   - CPU should query the byte 504 (S_CMD_SET) of EXT_CSD register by sending CMD8. If bit 4 is set to 1, then the device supports ATA mode.
   - If ATA mode is supported, the CPU should select the ATA mode by setting the ATA bit (bit 4) of the EXT_CSD register slice 191(CMD_SET) to activate the ATA command set for use. The CPU selects the command set using the SWITCH (CMD6) command.
   - The current mode selected is shown in byte 191 of the EXT_CSD register.If the device does not support ATA mode, then the device can be an MMC device or a CE-ATA v1.0 device.
   - Send RW_REG; if a response is received and the response data contains CE-ATA signature, the device is a CE-ATA device.
   - Otherwise the device is an MMC card.
6. You can change the card clock frequency after enumeration.

23.8.2.3 Clock programming

Clocking set up is done using register in Syscon, see Chapter 4 “LPC55S6x/LPC55S2x/LPC552x SYSCON”. The cclk_in has to be equal to the cclk_out. Therefore the registers that support this have to be:

- CLKDIV = 0x0 (bypass of clock divider).
- CLKSRC = 0x0
- CLKENA = 0x0 or 0x1. This register enables or disables clock for the card and enables low-power mode, which automatically stops the clock to a card when the card is idle for more than 8 clocks.

The Module loads each of these registers only when the START_CMD bit and the Update_clk_regs_only bit in the CMD register are set. When a command is successfully loaded, the Module clears this bit, unless the Module already has another command in the queue, at which point it gives an HLE (Hardware Locked Error); for details on HLEs, refer Section 23.8.2.19 “Error handling” Software should look for the START_CMD and the Update_clk_regs_only bits, and should also set the WAIT_PRVDATA_COMPLETE bit to ensure that clock parameters do not change during data transfer. Note that even though START_CMD is set for updating clock registers, the Module does not raise a command_done signal upon command completion.

23.8.2.4 No-Data command with or without response sequence

To send any non-data command, the software needs to program the CMD register and the CMDARG register with appropriate parameters. Using these two registers, the Module forms the command and sends it to the command bus. The Module reflects the errors in the command response through the error bits of the RINTSTS register.

When a response is received - either erroneous or valid - the Module sets the command_done bit in the RINTSTS register. A short response is copied in Response register 0 (RESP0), while a long response is copied to all four response registers. The

RESPONSE3 register bit 31 represents the MSB, and the RESPONSE0 register bit 0 represents the LSB of a long response.

For basic commands or non-data commands, follow these steps:

1. Program the Command register with the appropriate command argument parameter.

Program the Command register with the settings in Table 495.

2. Wait for command acceptance by CPU. The following happens when the command is loaded into the Module:
   - Module accepts the command for execution and clears the START_CMD bit in the CMD register, unless one command is in process, at which point the Module can load and keep the second command in the buffer.
   - If the Module is unable to load the command - that is, a command is already in progress, a second command is in the buffer, and a third command is attempted - then it generates an HLE (hardware-locked error).
   - Check if there is an HLE.
   - Wait for command execution to complete. After receiving either a response from a card or response time-out, the Module sets the command_done bit in the RINTSTS register. Software can either poll for this bit or respond to a generated interrupt.
   - Check if response_timeout error, response_CRC error, or response error is set. This can be done either by responding to an interrupt raised by these errors or by polling bits 1, 6, and 8 from the RINTSTS register. If no response error is received, then the response is valid. If required, the software can copy the response from the response registers.

Software should not modify clock parameters while a command is being executed.

Table 495. CMD register settings for No-Data command

Name	Value	Comment
START_CMD	1	
UPDATE_CLOCK_REGISTERS_ONLY	0	No clock parameters update command
DATA_EXPECTED	1	No data command.
SEND_INITIALIZATION	0	No clock parameters update command
STOP_ABORT_CMD	0	Can be 1 for commands to stop data transfer, such as CMD12
CMD_INDEX	Command index	
RESPONSE_LENGTH	0	Can be 1 for R2 (long) response
RESPONSE_EXPECT	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
User-selectable		

Table 495. CMD register settings for No-Data command *…continued*

Name	Value	Comment
USE_HOLD_REG	0	CMD and DATA sent to card bypassing HOLD register.
	1	CMD and DATA sent to card through the HOLD register. Hold settings applied through the SDIOCLKCTRL register in Syscon. See Section 4.5.68 “SDIO CCLKIN phase and delay control”
WAIT_PRVDATA_COMPLETE	1	Before sending command on command line, CPU should wait for completion of any data command in process, if any (recommended to always set this bit, unless the current command is to query status or stop data transfer when transfer is in progress)
CHECK_RESPONSE_CRC	1	0 – Do not check response CRC 1 – Check response CRC Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller.

23.8.2.5 Data transfer commands

Data transfer commands transfer data between the memory card and the Module. To send a data command, the Module needs a command argument, total data size, and block size. Software can receive or send data through the FIFO.

Before a data transfer command, software should confirm that the card is not busy and is in a transfer state, which can be done using the CMD13 and CMD7 commands, respectively.

For the data transfer commands, it is important that the same bus width that is programmed in the card should be set in the Card Type register (CTYPE).

The module generates an interrupt for different conditions during data transfer, which are reflected in the RINTSTS register as:

1. Data_Transfer_Over (bit 3) - When data transfer is over or terminated. If there is a response time-out error, then the Module does not attempt any data transfer and the *Data Transfer Over* bit is never set.
2. Transmit_FIFO_Data_request (bit 4) - FIFO threshold for transmitting data was reached; software is expected to write data, if available, in FIFO
3. Receive_FIFO_Data_request (bit 5) - FIFO threshold for receiving data was reached; software is expected to read data from FIFO.
4. Data starvation by CPU time-out (bit 10) - FIFO is empty during transmission or is full during reception. Unless software writes data for empty condition or reads data for full condition, the Module

   cannot continue with data transfer. The clock to the card has been stopped.
5. Data read time-out error (bit 9) - Card has not sent data within the time-out period.
6. Data CRC error (bit 7) - CRC error occurred during data reception.
7. Start bit error (bit 13) - Start bit was not received during data reception.
8. End bit error (bit 15) - End bit was not received during data reception or for a write operation; a CRC error is indicated by the card.

Conditions 6, 7, and 8 indicate that the received data may have errors. If there was a response

time-out, then no data transfer occurred.

23.8.2.6 Single-block or multiple-block read

Steps involved in a single-block or multiple-block read are:

1. Write the data size in bytes in the BYTCNT register.
2. Write the block size in bytes in the BLKSIZ register. The Module expects data from the card in blocks of size BLKSIZ each.
3. Program the CMDARG register with the data address of the beginning of a data read. Program the Command register with the parameters listed in Table 496. For SD and MMC cards, use CMD17 for a single-block read and CMD18 for a multiple-block read. For SDIO cards, use CMD53 for both single-block and multiple-block transfers.

   After writing to the CMD register, the Module starts executing the command; when the command is sent to the bus, the command_done interrupt is generated.
4. Software should look for data error interrupts; that is, bits 7, 9, 13, and 15 of the RINTSTS register. If required, software can terminate the data transfer by sending a STOP command.
5. Software should look for Receive_FIFO_Data_request and/or data starvation by CPU time-out conditions. In both cases, the software should read data from the FIFO and make space in the FIFO for receiving more data.
6. When a Data_Transfer_Over interrupt is received, the software should read the remaining data from the FIFO.

Table 496. CMD register settings for single-block or multiple-block read

Name	Value	Comment
START_CMD	1	
UPDATE_CLOCK_REGISTERS_ONLY	0	No clock parameters update command.
DATA_EXPECTED	1	Can be 1, but only for card reset commands, such as CMD0.
SEND_INITIALIZATION	0	No clock parameters update command.
STOP_ABORT_CMD	0	Can be 1 for commands to stop data transfer, such as CMD12.
SEND_AUTO_STOP	0/1	
TRANSFER_MODE	0	Block transfer.
READ_WRITE	1	Read to card.
CMD_INDEX	Command index	
RESPONSE_LENGTH	0	Can be 1 for R2 (long) response.
RESPONSE_EXPECT	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on.
User-selectable		

Table 496. CMD register settings for single-block or multiple-block read *…continued*

Name	Value	Comment
USE_HOLD_REG	0	CMD and DATA sent to card bypassing HOLD register.
	1	CMD and DATA sent to card through the HOLD register. Hold settings applied through the SDIOCLKCTRL register in Syscon. See Section 4.5.68 “SDIO CCLKIN phase and delay control”.
WAIT_PRVDATA_COMPLETE	1	Before sending command on command line, CPU should wait for completion of any data command in process, if any (recommended to always set this bit, unless the current command is to query status or stop data transfer when transfer is in progress).
CHECK_RESPONSE_CRC	1	0 – Do not check response CRC. 1 – Check response CRC. Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller.

23.8.2.7 Single-block or multiple-block write

Steps involved in a single-block or multiple-block write are:

1. Write the data size in bytes in the BYTCNT register.
2. Write the block size in bytes in the BLKSIZ register; the module sends data in blocks of size BLKSIZ each.
3. Program CMDARG register with the data address to which data should be written.
4. Write data in the FIFO; it is usually best to start filling data the full depth of the FIFO.
5. Program the Command register with the parameters listed in Table 497 For SD and MMC cards, use CMD24 for a single-block write and CMD25 for a multiple-block write. For SDIO cards, use CMD53 for both single-block and multiple-block transfers.

   After writing to the CMD register, module starts executing a command; when the command is sent to the bus, a command_done interrupt is generated.
6. Software should look for data error interrupts; that is, for bits 7, 9, and 15 of the RINTSTS register. If required, software can terminate the data transfer by sending the STOP command.
7. Software should look for Transmit_FIFO_Data_request and/or time-out conditions from data starvation by the CPU. In both cases, the software should write data into the FIFO.
8. When a Data_Transfer_Over interrupt is received, the data command is over. For an open-ended block transfer, if the byte count is 0, the software must send the STOP command. If the byte count is not 0, then upon completion of a transfer of a given number of bytes, the module should send the STOP command, if necessary. Completion of the AUTO-STOP command is reflected by the Auto_command_done interrupt - bit 14 of the RINTSTS register. A response to AUTO_STOP is stored in RESP1.

Table 497. CMD register settings for single-block or multiple-block writes

Name	Value	Comment
START_CMD	1	
UPDATE_CLOCK_REGISTERS_ONLY	0	No clock parameters update command.
DATA_EXPECTED	1	Can be 1, but only for card reset commands, such as CMD0.
SEND_INITIALIZATION	0	No clock parameters update command.
STOP_ABORT_CMD	0	Can be 1 for commands to stop data transfer, such as CMD12.
SEND_AUTO_STOP	0/1	
TRANSFER_MODE	0	Block transfer.
READ_WRITE	1	Write to card.
CMD_INDEX	Command index	
RESPONSE_LENGTH	0	Can be 1 for R2 (long) response.
RESPONSE_EXPECT	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on.
User-selectable		
USE_HOLD_REG	0	CMD and DATA sent to card bypassing HOLD register.
	1	CMD and DATA sent to card through the HOLD register. Hold settings applied through the SDIOCLKCTRL register in Syscon. See Section 4.5.68 “SDIO CCLKIN phase and delay control”
WAIT_PRVDATA_COMPLETE	1	Before sending command on command line, CPU should wait for completion of any data command in process, if any (recommended to always set this bit, unless the current command is to query status or stop data transfer when transfer is in progress)
CHECK_RESPONSE_CRC	1	0 – Do not check response CRC 1 – Check response CRC Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller.

23.8.2.8 Stream read

A stream read is like the block read mentioned in *Single-Block or Multiple-Block Read,* except for the following bits in the Command register (CMD):

- TRANSFER_MODE = 1; //Stream transfer
- CMD_INDEX = CMD20;

A stream transfer is allowed for only a single-bit bus width.

23.8.2.9 Stream write

A stream write is exactly like the block write mentioned in *Single-Block or Multiple-Block Write*, except for the following bits in the Command register (CMD):

- TRANSFER_MODE = 1;//Stream transfer
- CMD_INDEX = CMD11;

In a stream transfer, if the byte count is 0, then the software must send the STOP command. If the byte count is not 0, then when a given number of bytes completes a

transfer, the Module sends the STOP command. Completion of this AUTO_STOP command is reflected by the Auto_command_done interrupt. A response to an AUTO_STOP is stored in the RESP1 register.

A stream transfer is allowed for only a single-bit bus width.

23.8.2.10 Packed commands

In order to reduce overhead, read and write commands can be packed in groups of commands—either all read or all write—that transfer the data for all commands in the group in one transfer on the bus.

Packed commands can be of two types:

- Packed Write: CMD23 → CMD25.
- Packed Read: CMD23 → CMD25 → CMD23 → CMD18.

Packed commands are put in packets by the application software and are transparent to the core. For more information on packed commands, refer to the eMMC specification.

23.8.2.11 Sending Stop or Abort in middle of transfer

The STOP command can terminate a data transfer between a memory card and the Module, while the ABORT command can terminate an I/O data transfer for only the SDIO_IOONLY and SDIO_COMBO cards.

- Send STOP command - Can be sent on the command line while a data transfer is in progress; this command can be sent at any time during a data transfer. For information on sending this command, refer Section 23.8.2.4 “No-Data command with or without response sequence” .

  You can also use an additional setting for this command in order to set the Command register bits (5-0) to CMD12 and set bit 14 (STOP_ABORT_CMD) to 1. If stop STOP_ABORT_CMD is not set to 1, the user stopped a data transfer. Reset bit 13 of the Command register (WAIT_PRVDATA_COMPLETE) to 0 in order to make the Module send the command at once, even though there is a data transfer in progress.

  - Send ABORT command - Can be used with only an SDIO_IOONLY or SDIO_COMBO card. To abort the function that is transferring data, program the function number in ASx bits (CCCR register of card, address 0x06, bits (0-2) using CMD52.

    This is a non-data command. For information on sending this command, refer Section 23.8.2.4 “No-Data command with or without response sequence” .

    Program the CMDARG register with the appropriate command argument parameters listed in Table 498 Program the Command register using the command index as CMD52. Similar to the STOPcommand, set bit 14 of the Command register (STOP_ABORT_CMD) to 1, which must be done in order to inform the Module that the user aborted the data transfer. Reset bit 13 (WAIT_PRVDATA_COMPLETE) of the Command register to 0 in order to make the Module send the command at once, even though a data transfer is in progress.
  - Wait for command_transfer_over.
  - Check response (R5) for errors.

- During an open-ended card write operation, if the card clock is stopped because the FIFO is empty, the software must first fill the data into the FIFO and start the card clock before issuing a stop/abort command to the card.

Table 498. Parameters for CMDARG register

Bits	Contents	Value
31	R/W flag	1
30-28	Function number	0, for CCCR access
27	RAW flag	1, if needed to read after write
26	Don't care	-
25-9	Register address	0x06
8	Don't care	-
7-0	Write data	Function number to be aborted

23.8.2.12 Suspend or resume sequence

In an SDIO card, the data transfer between an I/O function and the Module can be temporarily halted using the SUSPEND command; this may be required in order to perform a high-priority data transfer with another function. When desired, the data transfer can be resumed using the RESUME command.

The following functions can be implemented by programming the appropriate bits in the CCCR register (function 0) of the SDIO card. To read from or write to the CCCR register, use the CMD52 command.

1. SUSPEND data transfer - Non-data command.
   - Check if the SDIO card supports the SUSPEND/RESUME protocol; this can be done through the SBS bit in the CCCR register of the card.

     Check if the data transfer for the required function number is in process; the function number that is currently active is reflected in bits 0-3 of the CCCR register. Note that if the BS bit is 1, then only the function number given by the FSx bits is valid.

     To suspend the transfer, set BR (bit 2) of the CCCR register.

     Poll for clear status of bits BR (bit 1) and BS (bit 0) of the CCCR. The BS (Bus Status) bit is 1 when the currently-selected function is using the data bus; the BR (Bus Release) bit remains 1 until the bus release is complete. When the BR and BS bits are 0, the data transfer from the selected function has been suspended.

     During a read-data transfer, the Module can be waiting for the data from the card. If the data transfer is a read from a card, then the Module must be informed after the successful completion of the SUSPEND command. The Module then resets the data state machine and comes out of the wait state. To accomplish this, set ABORT_READ_DATA (bit 8) in the control register (CTRL).

     Wait for data completion. Get pending bytes to transfer by reading the TCBCNT register.
2. RESUME data transfer - This is a data command.
   - Check that the card is not in a transfer state, which confirms that the bus is free for data transfer.

If the card is in a disconnect state, select it using CMD7. The card status can be retrieved in response to CMD52/CMD53 commands.

Check that a function to be resumed is ready for data transfer; this can be confirmed by reading the RFx flag in CCCR. If RF = 1, then the function is ready for data transfer.

To resume transfer, use CMD52 to write the function number at FSx bits (0-3) in the CCCR register. Form the command argument for CMD52 and write it in CMDARG; bit values are listed in Table 499.

- Write the block size in the BLKSIZ register; data will be transferred in units of this block size.

  Write the byte count in the BYTCNT register. This is the total size of the data; that is, the remaining bytes to be transferred. It is the responsibility of the software to handle the data.

  Program Command register; similar to a block transfer. For details, refer Section 23.8.2.6 “Single-block or multiple-block read” and Section 23.8.2.7 “Single-block or multiple-block write”.When the command register is programmed, the command is sent and the function resumes data transfer. Read the DF flag (resume Data Flag). If it is 1, then the function has data for the transfer and will begin a data transfer as soon as the function or memory is resumed. If it is 0, then the function has no data for the transfer.

  If the DF flag is 0, then in case of a read, the Module waits for data. After the data time-out period, it gives a data time-out error.

Table 499. Parameters for CMDARG register

Bits	Contents	Value
31	R/W flag	1
30-28	Function number	0, for CCCR access
27	RAW flag	1, read after write
26	Don’t care	-
25-9	Register address	0x0D
8	Don’t care	-
7-0	Write data	Function number to be aborted

23.8.2.13 Read_Wait sequence

Read_wait is used with only the SDIO card and can temporarily stall the data transfer-either from function or memory-and allow the CPU to send commands to any function within the SDIO device. The CPU can stall this transfer for as long as required. The Module provides the facility to signal this stall transfer to the card. The steps for doing this are:

1. Check if the card supports the read_wait facility; read SRW (bit 2) of the CCCR register. If this bit is 1, then all functions in the card support the read_wait facility. Use CMD52 to read this bit.
2. If the card supports the read_wait signal, then assert it by setting the READ_WAIT (bit 6) in the CTRL register.
3. Clear the READ_WAIT bit in the CTRL register.

23.8.2.14 CE-ATA data transfer commands

This section describes the CE-ATA data transfer commands. For information on the basic settings and interrupts generated for different conditions, refer Section 23.8.2.5 “Data transfer commands” .

23.8.2.14.1 Reset and device recovery

Before starting CE-ATA operations, the CPU should perform an MMC reset and initialization procedure. The CPU and device should negotiate the MMC TRAN state (defined by the MultiMedia Card System Specification) before the device enters the MMC TRAN state. The CPU should follow the existing MMC card enumeration procedure in order to negotiate the MMC

TRAN state. After completing normal MMC reset and initialization procedures, the CPU should query the initial ATA Task File values using RW_REG/CMD39.

By default, the MMC block size is 512 bytes-indicated by bits 1:0 of the srcControl register inside the CE-ATA device. The CPU can negotiate the use of a 1KB or 4KB MMC block size. The device indicates MMC block sizes that it can support through the srcCapabilities register; the CPU reads this register in order to negotiate the MMC block size. Negotiation is complete when the CPU controller writes the MMC block size into the srcControl register bits 1:0 of the device.

23.8.2.14.2 ATA task file transfer

ATA task file registers are mapped to addresses 0x00h-0x10h in the MMC register space. RW_REG is used to issue the ATA command, and the ATA task file is transmitted in a single RW_REG MMC command sequence.

The CPU software stack should write the task file image to the FIFO before setting the CMDARG and CMD registers. The CPU processor then sets the address and byte count in the CMDARG-offset 0x28 in the BIU register space-before setting the CMD (offset 0x2C) register bits.

For RW_REG, there is no command completion signal from the CE-ATA device

ATA Task File Transfer Using RW_MULTIPLE_REGISTER (RW_REG)

This command involves data transfer between the CE-ATA device and the Module. To send a data command, the Module needs a command argument, total data size, and block size. Software can receive or send data through the FIFO.

Table 500. Parameters for CMDARG register

Bits	Contents	Value
31	R/W flag	1 (write) or 0 (read)
30-24	Reserved	0
23:18	Starting register address for read/write; Dword aligned	0
17:16	Register address; Dword aligned	0
15-8	Reserved; bits cleared to 0 by CPU	0
7:2	Number of bytes to read/write; integral number of Dwords	16
1:0	Byte count in integral number of Dwords	0

- Program the Command (CMD) register as shown below.

Table 501. CMD register settings

Name	Value	Comment
START_CMD	1	
CSS_EXPECT	0	Command Completion Signal is not expected
READ_CEATA_DEVICE	0/1	1 – If RW_BLK or RW_REG read
UPDATE_CLOCK_REGISTERS_ONLY		No clock parameters update command
DATA_EXPECTED	1	
START_CMD	1	
CSS_EXPECT	0	Command Completion Signal is not expected
READ_CEATA_DEVICE	0/1	1 – If RW_BLK or RW_REG read
UPDATE_CLOCK_REGISTERS_ONLY	0	No clock parameters update command
DATA_EXPECTED	1	
SEND_INITIALIZATION	0	Can be 1, but only for card reset commands, such as CMD0
STOP_ABORT_CMD	0	
SEND_AUTO_STOP	0	
TRANSFER_MODE	0	Block transfer
READ_WRITE	0/1	0 - read from card 1 - Write to card
CMD_INDEX	Command index	
RESPONSE_LENGTH	0	
RESPONSE_EXPECT	1	
User-selectable		
USE_HOLD_REG	0	CMD and DATA sent to card bypassing HOLD register.
	1	CMD and DATA sent to card through the HOLD register. Hold settings applied through the SDIOCLKCTRL register in Syscon. See Section 4.5.68 “SDIO CCLKIN phase and delay control”
WAIT_PRVDATA_COMPLETE	1	0 – Sends command immediately 1 – Sends command after previous data transfer over
CHECK_RESPONSE_CRC	1	0 – Do not check response CRC 1 – Check response CRC

- Program the block size (BLKSIZ) register as shown below.

Table 502. BLKSIZ register

Bits	Value	Comment
31:16	0	Reserved bits as zeroes (0)
15:0	16	For accessing entire task file (16, 8-bit registers); block size of 16 bytes

- Program the Byte Count (BYTCNT) register as shown below.

Table 503. BYTCNT register

Bits	Value	Comment
31:0	16	For accessing entire task file(16, 8 bit registers); byte count value of 16 is used with the block size set to 16.

23.8.2.14.3 ATA payload transfer using RW_MULTIPLE_BLOCK (RW_BLK)

This command involves data transfer between the CE-ATA device and the Module. To send a data command, the Module needs a command argument, total data size, and block size. Software can receive or send data through the FIFO.

Steps involved in an ATA Task file transfer (read or write) are:

1. Write the data size in bytes in the BYTCNT register.
2. Write the block size in bytes in the BLKSIZ register; the Module expects a single block transfer.
3. Program the CMDARG register with the beginning register address.

You should program the CMDARG, CMD, BLKSIZ, and BYTCNT registers according to the following tables.

- Program the Command Argument (CMDARG) register as shown below.

Table 504. Parameters for CMDARG register

Bits	Contents	Value
31	R/W flag	0
30-24	Reserved	0
23:18	Starting register address for read/write; Dword aligned	0
17:16	Register address; Dword aligned	0
15-8	Reserved; bits cleared to 0 by CPU	0
7:2	Number of bytes to read/write; integral number of Dwords	16
1:0	Byte count in integral number of Dwords	0

- Program the Command (CMD) register as shown below.

Table 505. CMD register settings

Name	Value	Comment
START_CMD	1	
CSS_EXPECT	0	Command Completion Signal is not expected
READ_CEATA_DEVICE	0/1	1 – If RW_BLK or RW_REG read
UPDATE_CLOCK_REGISTERS_ONLY		No clock parameters update command
DATA_EXPECTED	1	
SEND_INITIALIZATION	0	Can be 1, but only for card reset commands, such as CMD0
STOP_ABORT_CMD	0	
SEND_AUTO_STOP	0	
TRANSFER_MODE	0	Block transfer
READ_WRITE	0/1	0 - read from card 1 - Write to card
CMD_INDEX	Command index	
RESPONSE_LENGTH	0	
RESPONSE_EXPECT	1	
User-selectable		

Table 505. CMD register settings *…continued*

Name	Value	Comment
USE_HOLD_REG	0	CMD and DATA sent to card bypassing HOLD register.
	1	CMD and DATA sent to card through the HOLD register. Hold settings applied through the SDIOCLKCTRL register in Syscon. See Section 4.5.68 “SDIO CCLKIN phase and delay control”
WAIT_PRVDATA_COMPLETE	1	0 – Sends command immediately 1 – Sends command after previous data transfer over
CHECK_RESPONSE_CRC	1	0 – Do not check response CRC 1 – Check response CRC

- Program the block size (BLKSIZ) register as shown in Table 506.

Table 506. BLKSIZ register

Bits	Value	Comment
31:16	0	Reserved bits as zeroes (0).
15:0	512, 1024, 4096	MMC block size can be 512, 1024, or 4096 bytes as negotiated by CPU.

- Program the Byte Count (BYTCNT) register as shown in Table 507.

Table 507. BYTCNT register

Bits	Value	Comment
31:0	N × block_size	byte_count should be integral multiple of block size; for ATA media access commands, byte count should be multiple of 4KB. (N × block_size = X × 4KB, where N and X are integers.

23.8.2.14.4 Sending command completion signal disable

While waiting for the Command Completion Signal (CCS) for an outstanding RW_BLK, the CPU can send a Command Completion Signal Disable (CCSD).

- Send CCSD - Module sends CCSD to the CE-ATA device if the SEND_CCSD bit is set in the CTRL register; this bit is set only after a response is received for the RW_BLK.
- Send internal Stop command - Send internally generated STOP (CMD12) command after sending the CCSD pattern. If SEND_AUTO_STOP_CCSD bit is also set when the controller is programmed to send the CCSD pattern, the Module sends the internally generated STOP command on the CMD line. After sending the STOP command, the Module sets the Auto Command Done bit in the RINTSTS register.

23.8.2.14.5 Recovery after command completion signal time-out

If time-out happened while waiting for Command Completion Signal (CCS), the CPU needs to send Command Completion Signal Disable (CCSD) followed by a STOP command to abort the pending ATA command. The CPU can program the Module to send internally generated STOP command after sending the CCSD pattern

- Send CCSD - Set the SEND_CCSD bit in the CTRL register.
- Reset bit 13 of the Command register (WAIT_PRVDATA_COMPLETE) to 0 in order to make the Module send the command at once, even though there is a data transfer in progress.

- Send internal STOP command - Set SEND_AUTO_STOP_CCSD bit in the CTRL register, which programs the CPU controller to send the internally generated STOP command. After sending the STOP command, the Module sets the Auto Command Done bit in the RINTSTS register.

23.8.2.14.6 Reduced ATA command set

It is necessary for the CE-ATA device to support the reduced ATA command subset. The following details discuss this reduced command set.

- IDENTIFY DEVICE - Returns 512-byte data structure to the CPU that describes device-specific information and capabilities. The CPU issues the IDENTIFY DEVICE command only if the MMC block size is set to 512-bytes; any other MMC block size has indeterminate results.

The CPU issues RW_REG for the ATA command, and the data is retrieved through RW_BLK.

The CPU controller uses the following settings while sending RW_REG for the IDENTIFY DEVICE ATA command. The following lists the primary bit

- CMD register setting - DATA_EXPECTED field set to 0.
- CMDARG register settings:
  - Bit [31] set to 0.
  - Bits [7:2] set to 128.
- Task file settings:
  - Command field of the ATA task file set to ECh.
  - Reserved fields of the task file cleared to 0.
- BLKSIZ register bits [15:0] and BYTCNT register - Set to 16

  The CPU controller uses the following settings for data retrieval (RW_BLK):\
- CMD register settings:
  - ccs_expect set to 1.
  - DATA_EXPECTED set to 1.
- CMDARG register settings:
  - Bit [31] set to 0 (Read operation).
  - Data Count set to 1 (16'h0001)
- BLKSIZ register bits [15:0] and BYTCNT register - Set to 512 IDENTIFY DEVICE can be aborted as a result of the CPU issued CMD12.
  - READ DMA EXT - Reads a number of logical blocks of data from the device using the Data-In data transfer protocol. The CPU uses RW_REG to issue the ATA command and RW_BLK for the data transfer.
  - WRITE DMA EXT - Writes a number of logical blocks of data to the device using the Data-Out data transfer protocol. The CPU uses RW_REG to issue the ATA command and RW_BLK for the data transfer.
  - STANDBY IMMEDIATE - No data transfer (RW_BLK) is expected for this ATA command, which causes the device to immediately enter the most aggressive power management mode that still retains internal device context.

- CMD register setting - DATA_EXPECTED field set to 0

  CMDARG register settings:

  - Bit [31] set to 1
  - Bits [7:2] set to 4
- Task file settings:
  - Command field of the ATA task file set to E0h.
  - Reserved fields of the task file cleared to 0.
- BLKSIZ register bits [15:0] and BYTCNT register - Set to 16
  - FLUSH CACHE EXT - No data transfer (RW_BLK) is expected for this ATA command. For devices that buffer/cache written data, the FLUSH CACHE EXT command ensures that buffered data is written to the device media. For devices that do not buffer written data, FLUSH CACHE EXT returns a success status. The CPU issues RW_REG for the ATA command, and the status is retrieved through CMD39/RW_REG; there can be error status for this ATA command, in which case fields other than the status field of the ATA task file are valid.
- The CPU uses the following settings while sending the RW_REG for STANDBY IMMEDIATE ATA command.
- CMD register setting - DATA_EXPECTED field set to 0
- CMDARG register settings
- Bit [31] set to 1
- Bits [7:2] set to 4
- Task file settings.
- Command field of the ATA task file set to EAh.
- Reserved fields of the task file cleared to 0.
- BLKSIZ register bits [15:0] and BYTCNT register - Set to 16.

23.8.2.15 Controller/DMA/FIFO reset usage

Communication with the card involves the following:

- Controller - Controls all functions of the module.
- FIFO - Holds data to be sent or received.
- DMA - If DMA transfer mode is enabled, then transfers data between system memory and the FIFO.
- Controller reset - Resets the controller by setting the CONTROLLER_RESET bit (bit 0) in the CTRL register; this resets the CIU and state machines, and also resets the BIU-to-CIU interface. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.
- FIFO reset - Resets the FIFO by setting the fifo_reset bit (bit 1) in the CTRL register; this resets the FIFO pointers and counters of the FIFO. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.

  DMA reset - Resets the internal DMA controller logic by setting the DMA_RESET bit (bit 2) in the CTRL register, which abruptly terminates any DMA transfer in process. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared

The following are recommended methods for issuing reset commands:

- Non-DMA transfer mode - Simultaneously sets CONTROLLER_RESET and FIFO_RESET; clears the RAWINTS register using another write in order to clear any resultant interrupt.
- Generic DMA mode - Simultaneously sets CONTROLLER_RESET, FIFO_RESET, and DMA_RESET; clears the RAWINTS register by using another write in order to clear any resultant interrupt. If a "graceful" completion of the DMA is required, then it is recommended to poll the STATUS register to see whether the dma request is 0 before resetting the DMA interface control and issuing an additional FIFO reset.
- In DMA transfer mode, even when the FIFO pointers are reset, if there is a DMA transfer in progress, it could push or pop data to or from the FIFO; the DMA itself completes correctly. In order to clear the FIFO, the software should issue an additional FIFO reset and clear any FIFO underrun or overrun errors in the RAWINTS register caused by the DMA transfers after the FIFO was reset.

23.8.2.16 Card read threshold

When an application needs to perform a Single or Multiple Block Read command, the application must program the CARDTHRCTL register with the appropriate Card Threshold size (CARDTHRESHOLD) and set the Card Read Threshold Enable (CARDRDTHREN) bit to 1'b1. This additional programming ensures that the Host controller sends a Read Command only if there is space equal to the CardRDThreshold available in the Rx FIFO. This in turn ensures that the card clock is not stopped in the middle a block of data being transmitted from the card. The Card Threshold can be set to the block size of the transfer, which ensures that there is a minimum of one block size of space in the RxFIFO before the controller enables the card clock.

The Card Read Threshold is required when the Round Trip Delay is greater than 0.5 cclk_in period as listed in Table 508.

Table 508. Card Read Threshold for Round Trip Delay

Model	Round Trip Delay (Delay_R = Delay_O + tODLY + Delay_I)	Is Stopping of Card Clock Allowed?	Card Read Threshold Required?
SDR25	Delay_R > 0.5cclk_in period	No	Yes
	Delay_R < 0.5cclk_in period	Yes	No
SDR12	Delay_R > 0.5cclk_in period	No	Yes
	Delay_R < 0.5cclk_in period	Yes	No

23.8.2.16.1 Recommended usage guidelines for card read threshold

1. The CARDTHRCTL register must be programmed before the programming the CMD register for a Data Read command.
2. The CARDTHRCTL register should not be programmed when a data transfer command is in progress.
3. A CardRdThreshold greater than or equal to the block size of the read transfer ensures that the card clock does not stop in between a block of data.
4. If the delay from the output of the card to the first sampling flip-flop in the Host controller is greater than 0.5 UI, then the Card Read Threshold must be enabled and the Card Threshold must be programmed as per guideline #3 to ensure that the Card Clock does not stop in between a block of data.

5. CardRdThreshold is greater than or equal to BlockSize (Recommended)The Card Read Threshold Size (CardRdThreshold) must be programmed to at the least 1*BlockSize of the multi block transfer to ensure that the Card Clock does not stop in between a block of data due to the RxFIFO becoming full during the read transfer.
6. CardRdThreshold is less than BlockSize

   the CarIfdReadThreshold is programmed to less than the BlockSize of the transfer, then the Host Controller System must ensure that the receive FIFO never becomes full and overflows during the read transfer; this can cause the card clock from the SDMMC/SDIO interface to stop. The SDMMC/SDIO interface is not be able to guarantee that the Card Clock does not stop during a read transfer.

Remark: If the CardRdThreshold, RX_WMARK, and MSIZE are programmed incorrectly, then the Card Clock may stop indefinitely and no interrupts will be generated from the Host Controller.

23.8.2.16.2 Card read threshold programming sequence

Most cards such as SDHC or SDXC typically support block sizes that are specified in the card or are fixed to 512 bytes. For SDIO cards—standard capacity SD cards that support READ_BL_PARTIAL = 1 and MMC cards—the block size is variable and can be chosen by the application.

In order to use the Card Read Threshold feature effectively and to guarantee that the Card Clock does not stop because of a FIFO Full condition in the middle of a block of data being read from the Card, follow these steps:

1. Choose block size

   The block size must be based on the following:

   – Rule 1 – DWORD-aligned Block Size

   The block size requested by the application from the card for the read transfer card must be DWORD-aligned.
2. Enable Card Read Threshold feature:

   – Rule 2 – Block Size ≤ Total Fifo Depth

   CardRdThreshold can be enabled only if the block size for the given transfer is less than the total depth of the FIFO.

   BlkSize ≤ FifoDepth

   Where:

   BlkSize = (block size in bytes) * 8/F_DATA_WIDTH; that is, the number of the block size in terms of FIFO locations

   FifoDepth = total number of FIFO Locations

   Remark: Note To use the Card Read Threshold for different block sizes when selecting the FIFO depth during configuration of the Host Controller, the selected FIFO depth must be greater than or equal to the maximum block size that the Host Controller is required to support. Typically, the largest block size to support is 512 bytes; the corresponding FIFO depth would 128 locations in the 32-bit-wide FIFO. If you choose a FIFO depth that is two or more times the maximum block size that the Host Controller is required to support, there will be greater performance and flexibility in choosing the MSIZE and RX_WMARK for the best throughput.

3. Choose CardRdThreshold:
   - If BlkSize ≥ ½ FifoDepth, choose CardRdThreshold such that CardRdThreshold ≤ BlkSize in bytes
   - If BlkSize < ½ FifoDepth, choose CardRdThreshold such that CardRdThreshold = BlkSize in bytes

   Remark: If the Host Controller is operating in Internal DMA Mode, or if the application does not use burst transfers to read data out of the FIFO while operating in External DMA mode or Slave mode, then use steps 4 and 5 below, and skip steps 6, 7, and 8. If the application uses burst transfers to read data out of the FIFO while operating in External DMA mode or in Slave mode, then skip steps 4 and 5, and instead follow steps 6, 7, and 8 below.
4. Choose DW_DMA_Mutiple_Transaction_Size:

   The possible values for the DW_DMA_Mutiple_Transaction_Size (MSIZE) are 1, 4, 8, 16, 32, 64, 128, and 256 transfers. Choose the value of MSIZE from the above transfer values so that MSIZE is a multiple of BlkSize.

   BlkSize% (MSIZE) = 0

   Special Cases:
   - When MSIZE = 1 transfer
     The MSIZE is equal to 1 if the block size chosen in Step 1 is not a multiple of the FIFO Width (in bytes).
     If MSIZE = 1 is not acceptable and a higher burst size is desired—that is, a higher MSIZE— then go back to Step 1 and recalculate the block size.
   - Internal DMA (IDMAC) mode
     The size of the data buffer (BuffSize in bytes) for each descriptor must be a multiple of MSIZE * H_DATA_WIDTH / 8. For example, BuffSize = n * MSIZE * H_DATA_WIDTH / 8, where n = 1, 2, 3…
5. Choose RX Watermark:
   - If MSIZE = 1, then the RX_WMARK = 1 or RX_WMARK = BlkSize -1

   Remark: Note For slave mode when the RX_WMARK is reached, the application must read only RX_WMARK number of locations from the FIFO.

   Additionally, for all DMA modes the RX Watermark (RX_WMARK) chosen must be a multiple of the chosen MSIZE.

   RX_WMARK = MSIZE *n, where n = 1, 2, 3 …

   Remark: If the Host Controller is operating in Internal DMA Mode, or if the application does not use burst transfers to read data out of the FIFO while operating in External DMA mode or Slave mode, then use steps 4 and 5 above, and skip steps 6, 7, and 8. If the application uses burst transfers to read data out of the FIFO while operating in External DMA mode or in Slave mode, then skip steps 4 and 5 above, and instead follow steps 6, 7, and 8 below.
6. Choose Burst Size:
   a. In order to determine burst transfers to drain data from the FIFO in external DMA or slave mode, use the following:
      BurstSize = number_of_beats * transfer_size in bytes per beat.

b. Choose the number of beats and the transfer size per beat so that BurstSize is a sub-multiple of BlkSize:
BlkSize % (BurstSize) = 0
Where:
BlkSize = Block Size in bytes * 8 /F_DATA_WIDTH
If H_DATA_WIDTH = 16 then:
(BlkSize * 2) % (BurstSize) = 0

7. Choose RX Watermark:
   - Use the following to determine RX Watermark (RX_WMARK):
     RX_WMARK = (BurstSize / F_DATA_WIDTH * 8) - 1

     Remark: Note For slave mode when the RX_WMARK is reached, the application must read only RX_WMARK number of locations from the FIFO.
8. Program MSIZE:
   - The possible values for the DW_DMA_Mutiple_Transaction_Size (MSIZE) are 1, 4, 8, 16, 32, 64, 128 or 256 transfers (refer FIFOTH[30:28]).

   a. Choose the following to determine the value of MSIZE:
   MSIZE > (BurstSize / H_DATA_WIDTH * 8)

23.8.2.16.3 Example card read threshold programming when BLKSIZE > 1/2 FIFO depth

Given:
H_DATA_WIDTH = 32
F_DATA_WIDTH = 32
Total FIFO Depth:
FifoDepth = 32 locations = 32 * F_DATA_WIDTH/8 = 128 bytes

The following shows how to program the CardRdThreshold when BlkSize is greater than ½ FifoDepth.

1. .Choose a DWORD-aligned BlkSize less than FifoDepth.

   If Block Size is 192 bytes, then BlkSize = 192 * 8/F_DATA_WIDTH = 48 FIFO locations
2. For DMA modes, choose MSIZE such that the BlkSize is a multiple of MSIZE.

   Possible MSIZE values 1, 4, 8 and 16, such that (48% MSIZE) = 0.

   Choose MSIZE = 8.
3. Choose the RX_WMARK = MSIZE -1.

   For example, RX_WMARK ≤ 8 - 1 = 7 FIFO locations.
4. Since BlkSize > ½ FifoDepth, choose CardRdThreshold = Block Size.

   CardRdThreshold = 192 bytes

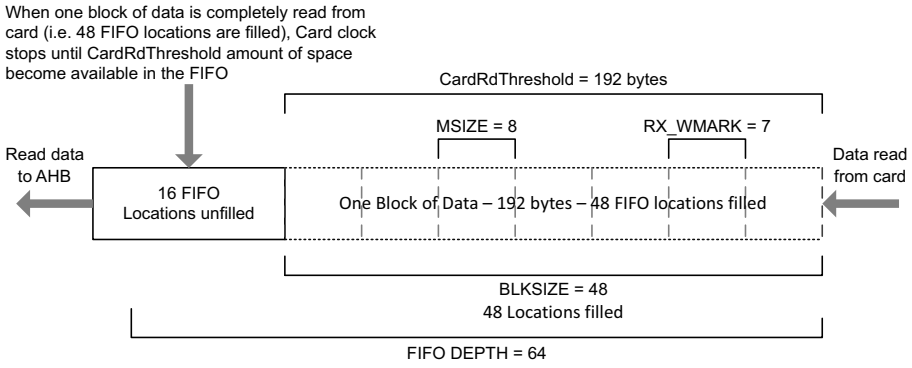


Fig 73. FIFO contents when BlkSize > ½ FifoDepth

23.8.2.16.4 Example card read threshold programming when BLKSIZE < 1/2 FIFO depth

Given:
H_DATA_WIDTH = 32
F_DATA_WIDTH = 32
Total FIFO Depth:
FifoDepth = 32 locations = 32 * F_DATA_WIDTH/8 = 128 bytes

The following shows how to program the CardRdThreshold when BlkSize is less than ½ FifoDepth.

1. Choose a DWORD-aligned BlkSize less than FifoDepth.

   If Block Size is 64 bytes, then BlkSize = 64 * 8/F_DATA_WIDTH = 64 * 8/32 = 16 FIFO locations
2. For DMA modes, choose MSIZE such that the BlkSize is a multiple of MSIZE.

   Possible MSIZE values are 1, 4, 8, and 16 so that (16%MSIZE) = 0.

   Choose MSIZE = 16.
3. Choose the RX_WMARK = MSIZE - 1.

   For example, RX_WMARK = 16 - 1 = 15 FIFO locations.
4. Since BlkSize < ½ FifoDepth, choose CardRdThreshold = Block Size.

   CardRdThreshold = 64 bytes.

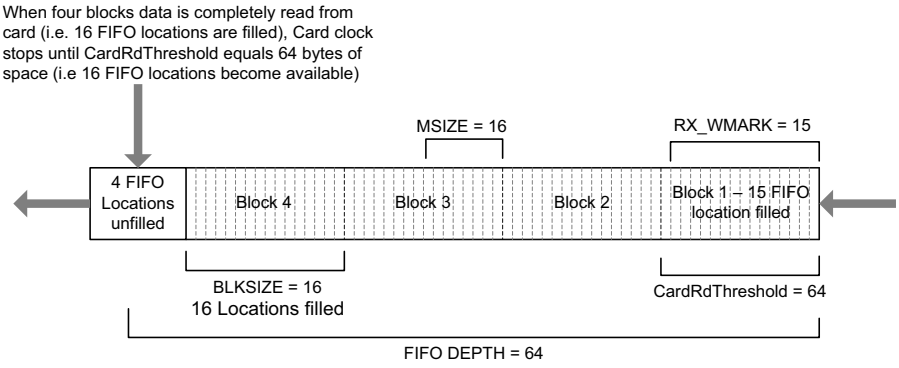


Fig 74. FIFO contents when BlkSize < ½ FifoDepth

23.8.2.17 Back-end power

The host controller needs to set the bit to enable the power sent to the back-end function of the card.

23.8.2.18 Master power control

Whether a card requires greater than 720 mW of power or not can be determined by reading the SMPC (Support Master Power Control) and the EMPC (Enable Master Power Control) registers in the card. Figure 75 illustrates the CCCR register.

12h	Power Control	RFU	RFU	RFU	RFU	RFU	EMPC	SMPC

Fig 75. Card common control registers (CCCR)

The chart in Figure 76 illustrates a basic flow that the device driver follows in order to detect the power requirements of the card.

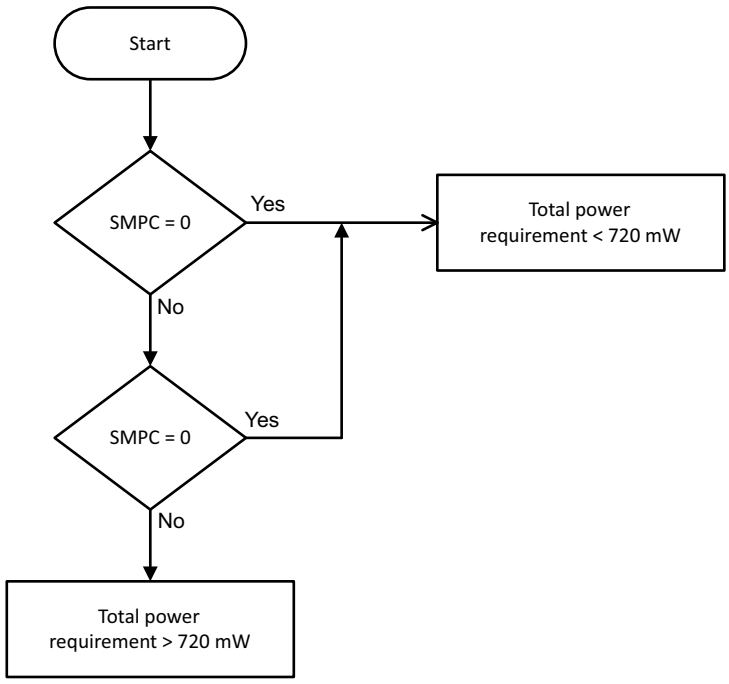


Fig 76. Flowchart for card power requirements

1. The driver reads the SMPC register.
2. If the value is 0, then the total power requirement is less than or equal to 720 mW.
3. If the value is 1, then the driver reads the EMPC register.
   a. If the value is 0 then the total power requirement is less than or equal to 720 mW.
   b. If the value is 1, then the total power requirement is greater than 720 mW.

   Once the power requirement has been determined to be more than 720 mW, the driver may tune the power to match the card requirements by tuning each function in the card.

   Figure 77 illustrates the two mechanisms for power tuning to get the correct power levels of each function.

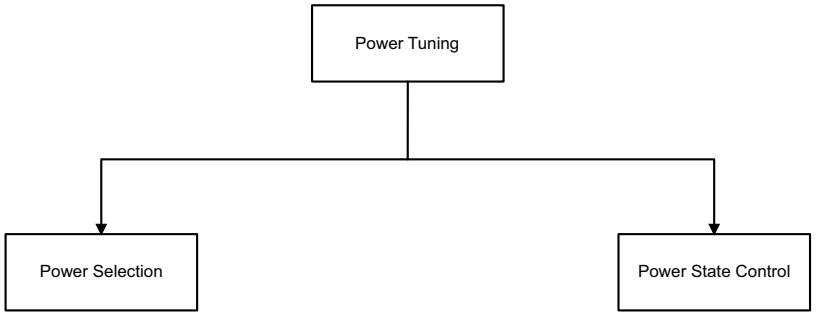


Fig 77. Different power tuning mechanisms

In addition to the CCCR, each supported I/O function has an FBR register.

Table 509. CMD register settings for single-block or multiple-block read

Addr: Field	Type	Description
SPS	R	Support Power Selection. • SPS = 1: Has two power modes selected by • EPSPS = 0: Has no Power Selection; EPS is zero. This bit is effective when EMPC = 1 in CCCR and PS[3:0] = 0.
EPS	R/W	Enable Power Selection • Indicates if function has Power Selection. • EPS = 0 (default): Operates in Higher Current Mode. Maximum current is given in TPLFE_HP_MAX_PWR_3.3V. • EPS = 1: Operates in Lower Current Mode. Maximum current is given in TPLFE_LP_MAX_PWR_3.3V. This bit is reset when IOEx = 0 and is effective when EMPC = 1 in CCR and PS[3:0] = 0.
PSx	R/W	Power State PS[3:0] If PS[3:0] is set to 0, TPL_CODE_CISTPL_FUNCE (22h) extension 01h is used and card power is controlled by EMPC and EPS (SDIO Ver 2.0-compatible). Power State control is defined by SDIO Ver 3.0 and is effective when EMPC is set to 1 and PS[3:0] is set to greater than 0. In this case, a list of card-supported power states is determined by TPL_CODE_CISTPL_FUNCE (22h) extension 02h (Power State Tuple). Host driver finds affordable power in Tuple and n (>0) is set to this field. Total current of card is sum of selected current of each function.

The chart in Figure 78 defines a basic flow that the device driver follows in order to detect and program the power requirements of each function in the card.

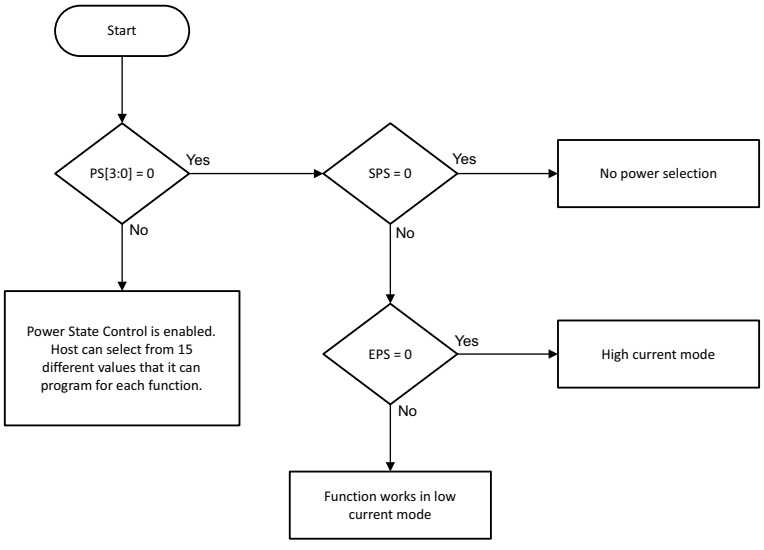


Fig 78. Flowchart for detect and program power requirements of functions

The driver reads the PS[3:0] register.

1. If the value of PS[3:0] is 1, the driver can program a value in the PS[3:0] register based on its affordable power in the tuples and program a number greater than 0.
2. If the value of PS [3:0] is 0, the driver reads the SPS register.

- If the value of SPS is 0, the function does not support power selection.
- If the value of SPS is 1, the driver reads the EPS register.
  If the value of EPS is 0, the function is set to High Current Mode.
  If the value of EPS is 1, the function is set to Low Current Mode.

Remark: Note Testing this feature may not be of much value to the verification team, but will help in building the driver.

23.8.2.19 Error handling

The Module implements error checking; errors are reflected in the RAWINTS register and can be communicated to the software through an interrupt, or the software can poll for these bits. Upon power-on, interrupts are disabled (INT_ENABLE in the CTRL register is 0), and all the interrupts are masked (bits 0-31 of the INTMASK register; default is 0). Error handling:

- Response and data time-out errors: For response time-out, software can retry the command. For data time-out, the Module has not received the data start bit - either for the first block or the intermediate block - within the time-out period, so software can either retry the whole data transfer again or retry from a specified block onwards. By reading the contents of the TCBCNT later, the software can decide how many bytes remain to be copied.
- Response errors: Set when an error is received during response reception. In this case, the response that copied in the response registers is invalid. Software can retry the command.
- Data errors: Set when error in data reception are observed; for example, data CRC, start bit not found, end bit not found, and so on. These errors could be set for any block-first block, intermediate block, or last block. On receipt of an error, the software can issue a STOP or ABORT command and retry the command for either whole data or partial data.
- Hardware locked error: Set when the Module cannot load a command issued by software. When software sets the START_CMD bit in the CMD register, the Module tries to load the command. If the command buffer is already filled with a command, this error is raised. The software then has to reload the command.
- FIFO underrun/overrun error: If the FIFO is full and software tries to write data in the FIFO, then an overrun error is set. Conversely, if the FIFO is empty and the software tries to read data from the FIFO, an underrun error is set. Before reading or writing data in the FIFO, the software should read.
- Data starvation by CPU time-out: Raised when the Module is waiting for software intervention to transfer the data to or from the FIFO, but the software does not transfer within the stipulated time-out period. Under this condition and when a read transfer is in process, the software.
- Should read data from the FIFO and create space for further data reception. When a transmit operation is in process, the software should fill data in the FIFO in order to start transferring data to the card.
- CRC Error on Command: If a CRC error is detected for a command, the CE-ATA device does not send a response, and a response time-out is expected from the module. The ATA layer is notified that an MMC transport layer error occurred.

- Write operation: Any MMC Transport layer error known to the device causes an outstanding ATA command to be terminated. The ERR bits are set in the ATA status registers and the appropriate error code is sent to the ATA Error register.
- If nIEN = 0, then the Command Completion Signal (CCS) is sent to the CPU.

  If device interrupts are not enabled (nIEN = 1), then the device completes the entire Data Unit Count if the CPU controller does not abort the ongoing transfer.

  During a multiple-block data transfer, if a negative CRC status is received from the device, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the RINTSTS register. It then continues further data transmission until all the bytes are transmitted.
- Read operation: If MMC transport layer errors are detected by the CPU controller, the CPU completes the ATA command with an error status.

The CPU controller can issue a Command Completion Signal Disable (CCSD) followed by a STOP TRANSMISSION (CMD12) to abort the read transfer. The CPU can also transfer the entire Data Unit Count bytes without aborting the data transfer.

23.8.2.20 Transmission and reception with internal DMAC (IDMAC)

The general sequence of events for transmit and receive is as follows:

1. Program the required programming in the Bus Mode register (BMOD). If IDMAC enable is disabled during the middle of an IDMAC transfer, it has no effect. It will only take effect for a new data transfer command. Issuing a Software reset will immediately terminate the transfer. It is recommended that the host issue a reset to DMA interface by setting DMA_RESET bit of the CTRL register and then issue a IDMAC software reset. The PBL contents are read-only and a direct reflection of the DW_DMA_Mutiple_Transaction_Size contents in FIFOTH register. The Fixed burst bit has to be programmed appropriately for system performance.
2. In case a Descriptor Unavailable Interrupt is asserted, the host needs to form the descriptor, appropriately set its own bit and then write to Poll Demand register (PLDMND) for the IDMAC to re-fetch the descriptor.
3. It is always appropriate for the host to enable abnormal interrupts since any errors related to the transfer are reported to the host.
4. For handling scenarios like Fatal Bus Error, Abort and FIFO overrun/under-run refer Section 23.8.3.2.6 “Interrupts”.

   For more information, refer Section 23.8.3.2.4 “Transmission” and Section 23.8.3.2.5 “Reception”.

23.8.3 DMA descriptors

The SD/MMC DMA controller uses the following descriptor structures:

- Dual buffer Structure – The distance between two descriptors is determined by the Skip Length value programmed in the Descriptor Skip Length (DSL) field of the Bus Mode register (BMOD).
- Chain Structure – Each descriptor points to a unique buffer and the next descriptor.

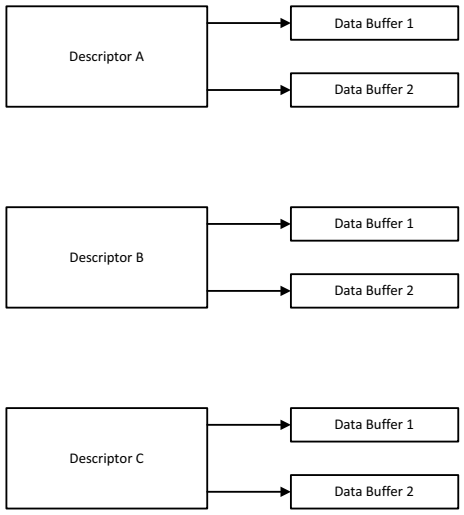


Fig 79. Dual buffer descriptor structure

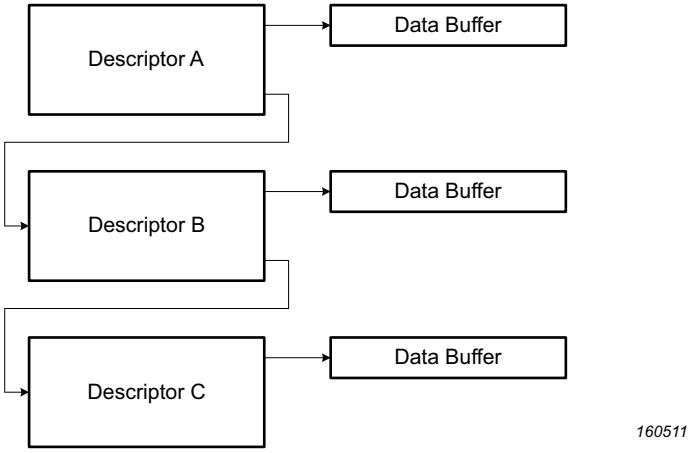


Fig 80. Chain descriptor structure

23.8.3.1 SD/MMC DMA descriptors

23.8.3.1.1 SD/MMC DMA descriptor DESC0

The DES0 descriptor contains control and status information.

Table 510. SD/MMC DMA DESC0 descriptor

Bit	Symbol	Description
0	-	Reserved.
1	DIC	Disable interrupt on completion When set, this bit will prevent the setting of the TI/RI bit of the IDMAC Status register (IDSTS) for the data that ends in the buffer pointed to by this descriptor.
2	LD	Last descriptor When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the data.

Table 510. SD/MMC DMA DESC0 descriptor *…continued*

Bit	Symbol	Description
3	FS	First descriptor When set, this bit indicates that this descriptor contains the first buffer of the data. If the size of the first buffer is 0, next Descriptor contains the beginning of the data.
4	CH	Second address chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When this bit is set, BS2 (DES1[25:13]) should be all zeros.
5	ER	End of ring When set, this bit indicates that the descriptor list reached its final descriptor. The IDMAC returns to the base address of the list, creating a Descriptor Ring. This is meaningful for only a dual buffer descriptor structure.
29:6	-	Reserved
30	CES	Card error summary These error bits indicate the status of the transaction to or from the card. These bits are also present in RINTSTS Indicates the logical OR of the following bits: • EBE: End Bit Error • RTO: Response Time-out • RCRC: Response CRC • SBE: Start Bit Error • DRTO: Data Read Time-out • DCRC: Data CRC for Receive • RE: Response Error
31	OWN	When set, this bit indicates that the descriptor is owned by the SD/MMC DMA. When this bit is reset, it indicates that the descriptor is owned by the Host. The SD/MMC DMA clears this bit when it completes the data transfer.

23.8.3.1.2 SD/MMC DMA descriptor DESC1

The DES1 descriptor contains the buffer size.

Table 511. SD/MMC DMA DESC1 descriptor

Bit	Symbol	Description
12:0	BS1	Buffer 1 Size Indicates the data buffer byte size, which must be a multiple of 4 bytes. The buffer 1 size must not exceed 4096 bytes (0x1000). If this field is 0, the DMA ignores this buffer and proceeds to the next descriptor in case of a chain structure, or to the next buffer in case of a dual buffer structure. **Remark:** If there is only one descriptor and only one buffer to be programmed, use only the Buffer 1 and not Buffer 2.
25:13	BS2	Buffer 2 Size These bits indicate the second data buffer byte size. The buffer size must be a multiple of 4. The buffer 2 size must not exceed 4096 bytes (0x1000).This field is not valid if DES0[4] is set.
31:26	-	Reserved

23.8.3.1.3 SD/MMC DMA descriptor DESC2

The DES2 descriptor contains the address pointer to the data buffer.

Table 512. SD/MMC DMA DESC2 descriptor

Bit	Symbol	Description
31:0	BAP1	Buffer address pointer 1 These bits indicate the physical address of the first data buffer. The SD/MMC DMA ignores DES2 [1:0], corresponding to the bus width of 64/32/16, internally.

23.8.3.1.4 SD/MMC DMA descriptor DESC3

The DES3 descriptor contains the address pointer to the next descriptor if the present descriptor is not the last descriptor in a chained descriptor structure or the second buffer address for a dual buffer structure.

Table 513. SD/MMC DMA DESC3 descriptor

Bit	Symbol	Description
31:0	BAP2	Buffer address pointer 2/ Next descriptor address These bits indicate the physical address of the second buffer when the dual buffer structure is used. If the Second Address Chained (DES0[4]) bit is set, then this address contains the pointer to the physical memory where the Next Descriptor is present. If this is not the last descriptor, then the Next Descriptor address pointer must be bus-width aligned (DES3[1:0] = 0, internally the LSBs are ignored).

23.8.3.2 Initialization

For the SD/MMC DMA initialization, follow these steps:

1. Write to the Bus Mode register (BMOD) to set the Host bus access parameters.
2. Write to the Interrupt Enable register (IDINTEN) to mask unnecessary interrupt causes.
3. The software driver creates either the Transmit or the Receive descriptor list. Then it writes to Descriptor List Base Address register (DBADDR), providing the IDMAC with the starting address of the list.
4. The SD/MMC DMA engine attempts to acquire descriptors from the descriptor lists.

23.8.3.2.1 Host bus burst access

The SD/MMC DMA attempts to execute fixed-length burst transfers on the AHB Master interface if configured using the FB bit of the IDMAC Bus Mode register. The maximum burst length is indicated and limited by the PBL field. The descriptors are always accessed in the maximum possible burst-size for the 16-bytes to be read: 16*8/bus-width.

The SD/MMC DMA initiates a data transfer only when sufficient space to accommodate the configured burst is available in the FIFO or the number of bytes to the end of data, when less than the configured burst-length. The SD/MMC DMA indicates the start address and the number of transfers required to the AHB Master Interface. When the AHB Interface is configured for fixed-length bursts, then it transfers data using the best combination of INCR4/8/16 and SINGLE transactions. Otherwise, in no fixed-length bursts, it transfers data using INCR (undefined length) and SINGLE transactions.

23.8.3.2.2 Host data buffer alignment

The transmit and receive data buffers in host memory must be 32-bit aligned.

23.8.3.2.3 Buffer size calculations

The driver knows the amount of data to transmit or receive. For transmitting to the card, the IDMAC transfers the exact number of bytes to the FIFO, indicated by the buffer size field of DES1.

If a descriptor is not marked as last - LS bit of DES0 - then the corresponding buffers of the descriptor are full, and the amount of valid data in a buffer is accurately indicated by its buffer size field. If a descriptor is marked as last, then the buffer cannot be full, as indicated by the buffer size in DES1. The driver is aware of the number of locations that are valid in this case.

23.8.3.2.4 Transmission

The SD/MMC transmission occurs as follows:

1. The Host sets up the Descriptor (DES0-DES3) for transmission and sets the OWN bit (DES0[31]). The Host also prepares the data buffer.
2. The Host programs the write data command in the CMD register in BIU.
3. The Host will also program the required transmit threshold level (TX_WMARK field in FIFOTH register).
4. The SD/MMC DMA determines that a write data transfer needs to be done as a consequence of step 2.
5. The SD/MMC DMA engine fetches the descriptor and checks the OWN bit. If the OWN bit is not set, it means that the host owns the descriptor. In this case the SD/MMC DMA enters suspend state and asserts the Descriptor Unable interrupt in the SD/MMC DMA status register (IDSTS). In such a case, the host needs to release the SD/MMC DMA by writing any value to the Poll Demand register (PLDMND).
6. It will then wait for Command Done (CD) bit and no errors from BIU which indicates that a transfer can be done.
7. The SD/MMC DMA engine will now wait for a DMA interface request from BIU. This request will be generated based on the programmed transmit threshold value. For the last bytes of data which can’t be accessed using a burst, SINGLE transfers are performed on AHB Master Interface.
8. The SD/MMC DMA fetches the Transmit data from the data buffer in the Host memory and transfers to the FIFO for transmission to card.
9. When data spans across multiple descriptors, the SD/MMC DMA will fetch the next descriptor and continue with its operation with the next descriptor. The Last Descriptor bit in the descriptor indicates whether the data spans multiple descriptors or not.
10. When data transmission is complete, status information is updated in SD/MMC DMA status register (IDSTS) by setting Transmit Interrupt, if enabled. Also, the OWN bit is cleared by the SD/MMC DMA by performing a write transaction to DES0.

23.8.3.2.5 Reception

The SD/MMC reception occurs as follows:

1. The Host sets up the Descriptor (DES0-DES3) for reception, sets the OWN (DES0[31]).
2. The Host programs the read data command in the CMD register in BIU.

3. The Host will program the required receive threshold level (RX_WMARK field in FIFOTH register).
4. The SD/MMC DMA determines that a read data transfer needs to be done as a consequence of step 2.
5. The SD/MMC DMA engine fetches the descriptor and checks the OWN bit. If the OWN bit is not set, it means that the host owns the descriptor. In this case the DMA enters suspend state and asserts the Descriptor Unable interrupt in the SD/MMC DMA Status register (IDSTS). In such a case, the host needs to release the SD/MMC DMA by writing any value to the Poll Demand register (PLDMND).
6. It will then wait for Command Done (CD) bit and no errors from BIU which indicates that a transfer can be done.
7. The SD/MMC DMA engine will now wait for a DMA interface request (dw_dma_req) from BIU. This request will be generated based on the programmed receive threshold value. For the last bytes of data which can't be accessed using a burst, SINGLE transfers are performed on AHB.
8. The SD/MMC DMA fetches the data from the FIFO and transfer to Host memory.
9. When data spans across multiple descriptors, the SD/MMC DMA will fetch the next descriptor and continue with its operation with the next descriptor. The Last Descriptor bit in the descriptor indicates whether the data spans multiple descriptors or not.
10. When data reception is complete, status information is updated in SD/MMC DMA Status register (IDSTS) by setting Receive Interrupt, if enabled. Also, the OWN bit is cleared by the SD/MMC DMA by performing a write transaction to DES0.

23.8.3.2.6 Interrupts

Interrupts can be generated as a result of various events. The SD/MMC DMA Status register (IDSTS) contains all the bits that might cause an interrupt. The SD/MMC DMA Interrupt Enable register (IDINTEN) contains an Enable bit for each of the events that can cause an interrupt.

There are two groups of summary interrupts - Normal and Abnormal - as outlined in Status register (IDSTS). Interrupts are cleared by writing a 1 to the corresponding bit position. When all the enabled interrupts within a group are cleared, the corresponding summary bit is cleared. When both the summary bits are cleared, the interrupt signal is deasserted.

Interrupts are not queued and if the interrupt event occurs before the driver has responded to it, no additional interrupts are generated. For example, Receive Interrupt (IDSTS[1]) indicates that one or more data was transferred to the Host buffer.

An interrupt is generated only once for simultaneous, multiple events. The driver must scan the SD/MMC DMA Status register for the interrupt cause.

23.8.3.2.7 Abort

When the host issues CMD12 when a data transfer on the card data lines is in progress, the FSM closes the present descriptor after completing the transfer of data until a DTO interrupt is asserted. Once an abort command is issued, the DMA performs single burst transfers:

1. When the host issues CMD12 when a data transfer on the card data lines is in progress, the FSM closes the present descriptor after completing the transfer of data until a DTO interrupt is asserted. Once an abort command is issued, the DMAC performs single burst transfers.
2. For a card read, the SD/MMC DMA keeps popping data from FIFO and writes to the host memory until a DTO interrupt is generated. This is required since DTO interrupt is not generated until and unless all the FIFO data is emptied.

Remark: The following scenarios apply for closing the descriptors:

- In case of an FBE, the current descriptor and the remaining unread descriptors are not closed by the SD/MMC DMA.
- In case of a write abort, only the current descriptor during which an abort occurred is closed by the SD/MMC DMA. The remaining unread descriptors are not closed by the IDMAC.
- In case of a read abort, the SD/MMC DMA pops the data out of the FIFO and writes them to the corresponding descriptor data buffers. The remaining unread descriptors are not closed.

23.8.3.2.8 FBE scenarios

An FBE occurs due to an AHB error response on the AHB bus. This is a system error, so the software driver should not perform any further programming to the SD/MMC The only recovery mechanism from such scenarios is to do one of the following:

- Issue a hard reset by asserting the reset_n signal.
- Do a program controller reset by writing to the CTRL[0] register.

23.8.3.2.9 FIFO overflow and underflow

During normal data transfer conditions, FIFO overflow and underflow will not occur. However if there is a programming error, then FIFO overflow/underflow can result. For example, consider the following scenarios.

For transmit: PBL = 4, Tx watermark = 1. For these programming values, if the FIFO has only one location empty, it issues a dw_dma_req to DMA state machine. Due to PBL value = 4, the DMA performs 4 pushes into the FIFO. This will result in a FIFO overflow interrupt.

For receive: PBL=4, Rx watermark = 1. For these programming values, if the FIFO has only one location filled, it issues a dw_dma_req to the DMA state machine. Due to PBL value = 4, the DMA performs 4 pops to the FIFO. This will result in a FIFO underflow interrupt.

The driver should ensure that the number of bytes to be transferred as indicated in the descriptor should be a multiple of 4 bytes. For example, if the BYTCNT = 13, the number of bytes indicated in the descriptor should be 16.

23.8.3.2.10 Programming of PBL and watermark levels

The SD/MMC DMA performs data transfers depending on the programmed PBL and threshold values.Table 514 lists the allowed programming values.

Table 514. PBL and watermark levels

PBL (number of transfers)	Transmit/receive watermark value
1	greater than or equal to 1
4	greater than or equal to 4
8	greater than or equal to 8
16	greater than or equal to 16
32	greater than or equal to 64

23.8.4 Back-end power

Each device needs one bit to control the back-end power supply for an embedded device; this bit does not control the VDDH of the host controller. A BACK_END_POWER register enables software programming for back-end power. The value on this register is output to the back_end_power signal, which can be used to switch power on and off the embedded device.

23.8.5 Master power control

SDIO cards prior to the 1.10 specification required a maximum power of 200 mA * 3.3v = 720 mW, regardless of the number of functions in each card. Newer cards have increased requirements greater than 720 mW.

23.8.6 Dedicated interrupt pin

The interrupt line is defined only for eSDIO devices. This interrupt line can operate even when the card clock is switched off and can be used only during an asynchronous interrupt period. Figure 81 shows the pins of an eSDIO device.

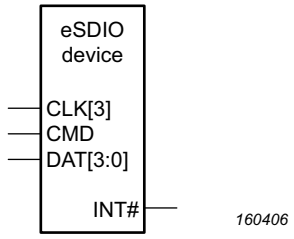


Fig 81. eSDIO device

23.8.7 Card-Detect and Write-Protect mechanism

Figure 82 illustrates how the SDMMC/SDIO interface card detection and write-protect signals are connected. Most of the SD_MMC sockets have card-detect pins. When no card is present, SD_CARD_DETECT_N is 1 due to the pull-up. When the SD_MMC card is inserted, the card-detect pin is shorted to ground, which makes card_detect_n go to 0. Similarly in SD cards, when the write-protect switch is toward the left, it shorts the SD_WR_PRT port to ground.

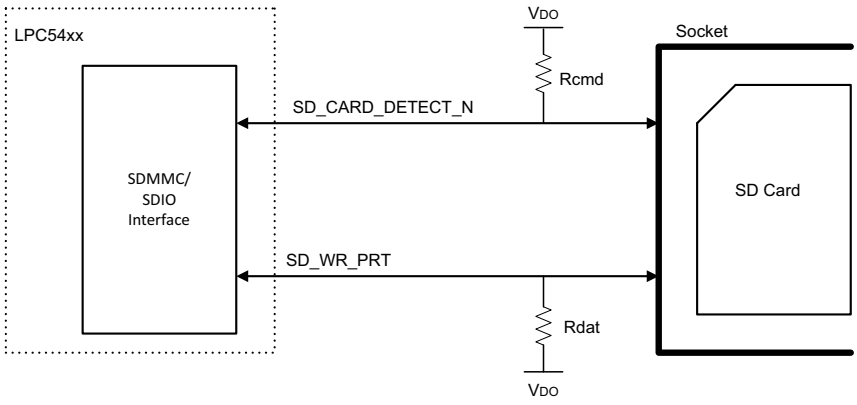


Fig 82. Card-Detect and Write-Protect

23.8.8 Termination requirement

Figure 84 illustrates the SDMMC/SDIO interface termination requirements, which is required to pull up the SD_CMD and SD_Dn lines on the SD_MMC_CEATA bus. The recommended specification for pull-up on the SD_CMD line is 4.7K - 100K for MMC, and 10K - 100K for an SD. The recommended pull-up on the SD_Dn line is 50K - 100K.

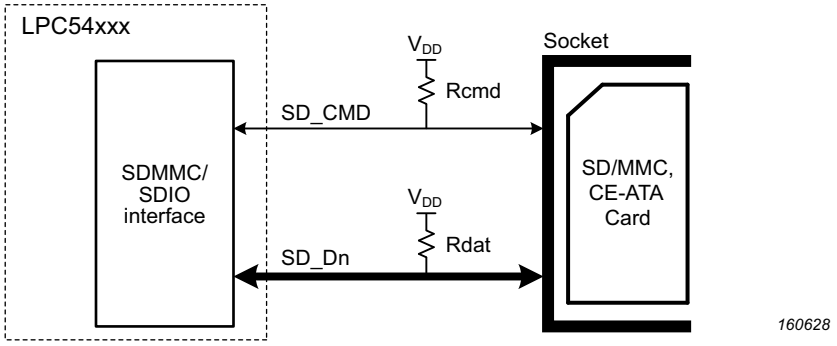


Fig 83. Termination

23.8.9 Rcmd and Rod calculation

The SD and MMC card enumeration happens at a very low frequency – 100-400 kHz. During enumeration open-drive mode is used. The pull-up in the command line pulls the bus to 1 when the card drives “z.” MMC interrupt mode also uses the pull-up. During normal data transfer, the card driver switches to push-pull mode.

For example, if enumeration is done at 400 kHz and the total bus capacitance is 20 pf, the pull-up needed during enumeration is:

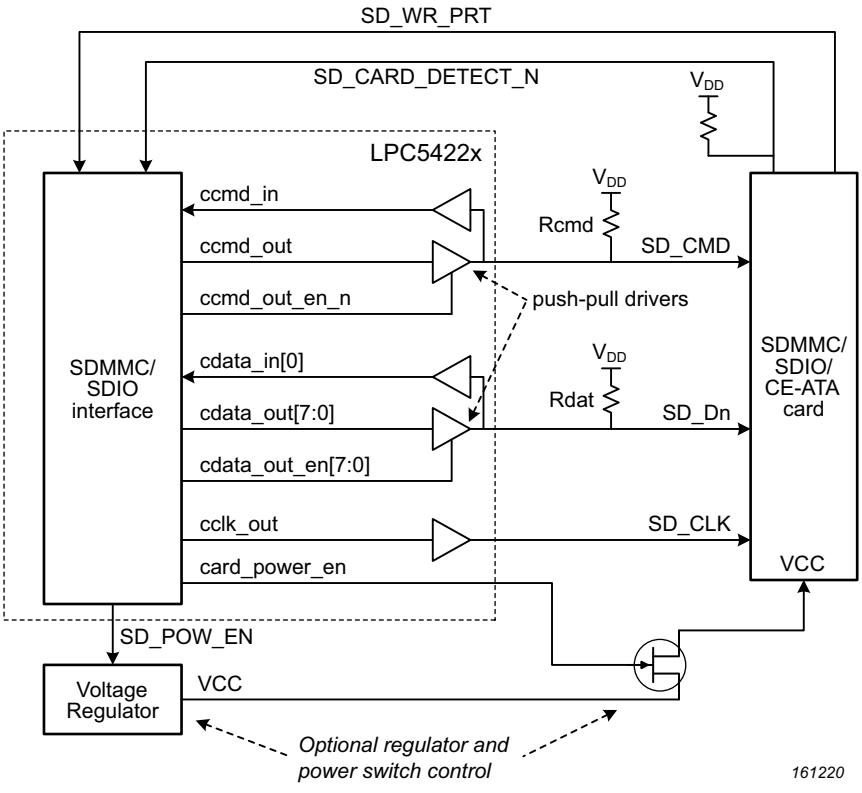
2.2 RC = rise-time = 1/400 kHz
R = 1/(2.2 * C * 100 kHz
== 1/(2.2 x 20 x 10**-12 x 400 x 10**3
== 1/(1.76 x 10**-5
== 56.8K

The Rod and Rcmd should be adjusted in such a way that the effective pull-up is at the maximum 5.68K during enumeration. Since only one card is supported, a fixed Rcmd resister is sufficient and there is no need for an additional Rod pull-up during enumeration. You should also ensure the effective pull-up will not violate the Iol rating of the drivers.

23.8.10 Interfacing to SD memory, SDIO, and MMC card

Figure 84 illustrates the connection between the SDMMC/SDIO interface and SD memory, SDIO, and MMC cards in SD_MMC_CE-ATA mode, which supports all three types of cards in the same controller. The primary differences between the MMC-Ver3.3-only and SD_MMC_CE-ATA modes are:

- In SD_MMC_CE-ATA mode, an SD card can be either a 1-bit data or 4-bit data card; MMC (3.31) cards are always 1-bit only, while MMC (4.0) cards could be either in 1-bit, 4-bit, or 8-bit mode.



Rcmd - typically 4.7 to 100 K
Rdat - typically 50 to 100 K
Rod - VDD / (Rod || Rcmd) < Iol (2 mA)

Fig 84. SD/MMC, SDIO, and MMC Card/CE-ATA single-card connection

23.9 Clocking and timing guidelines

The SDMMC/SDIO interface (also referred to as the host controller) has four input clocks and one output clock.

23.9.1 Clock domains

The SDMMC/SDIO interface has the following clocks:

Table 515. Clocks

Clock name	Input/Output	Edge used within the controller
clk	Input	Rising edge.
cclk_in	Input	Rising and falling edges.
cclk_in_drv	Input	Rising and falling edges.
cclk_in_sample	Input	Rising and falling edges.
cclk_out	Output	-

23.9.1.1 Relationships between clocks

Figure 85 shows the different clocks in the SDMMC/SDIO interface:

- cclk_in
- cclk_out
- cclk_in_sample
- cclk_in_drv

Remark: Note that the AMBA clock should be at least equal to or greater than one-tenth of cclk_in.

All relevant delays corresponding to the core are indicated—bypass_mode refers to the selection where cclk_in clock divider logic can be bypassed; that is, cclk_out is an undivided version of cclk_in.

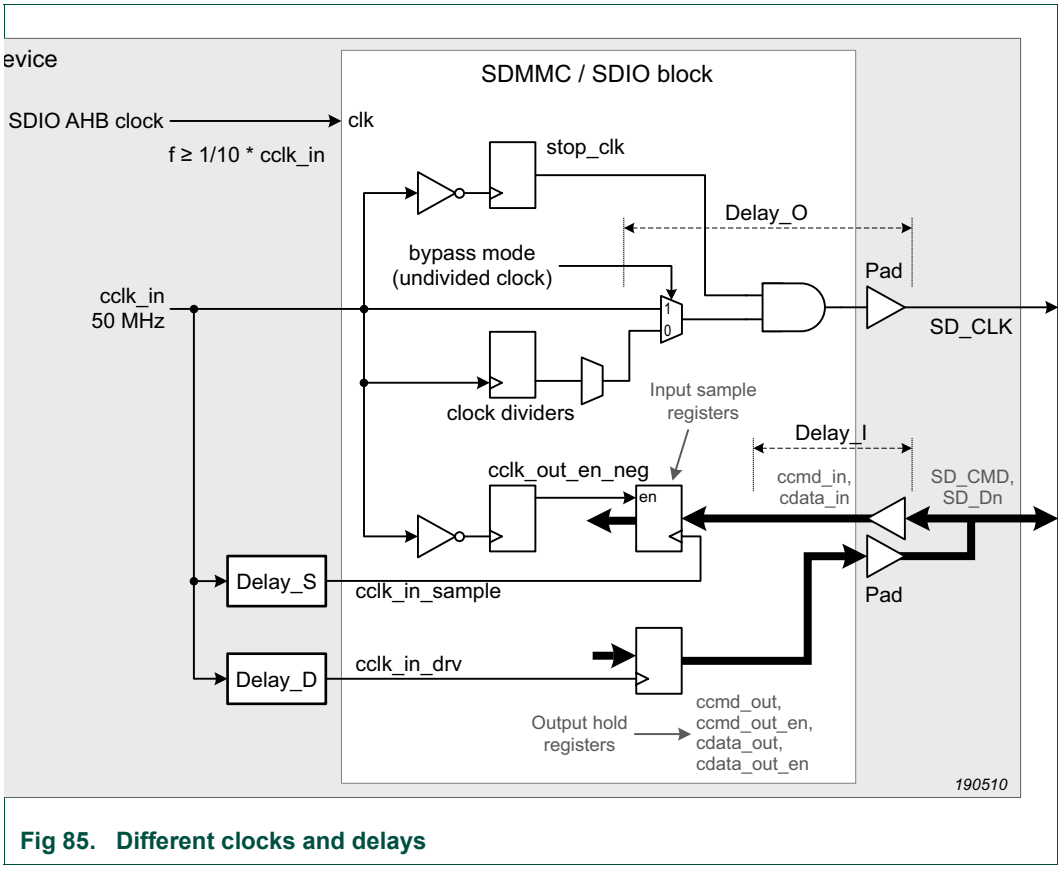


Fig 85. Different clocks and delays

Definitions:

- Delay_O = cclk_in to cclk_out delay (including PAD).
- Delay_I = Input PAD delay + routing delay to input register.
- tODLY = cclk_out to card output delay (varies across card manufactures and speed modes).
- Delay_S = Delay by which cclk_in_sample is phase-shifted with regard to cclk_in.
- Delay_D = Delay by which cclk_in_drv is phase-shifted with regard to cclk_in.
- Delay_R = Delay_O+Delay_I + tODLY = Total turn-around delay

Figure 86 illustrates relationships for these delays. Figure 87 shows the relationships between cclk_in, cclk_out, cclk_in_sample, and cclk_in_drv.

Remark: Controller outputs driven to the card are driven on cclk_in_drv when USE_HOLD_REG = 1. SDMMC/SDIO interface sampled_inputs are sampled on cclk_in_sample.

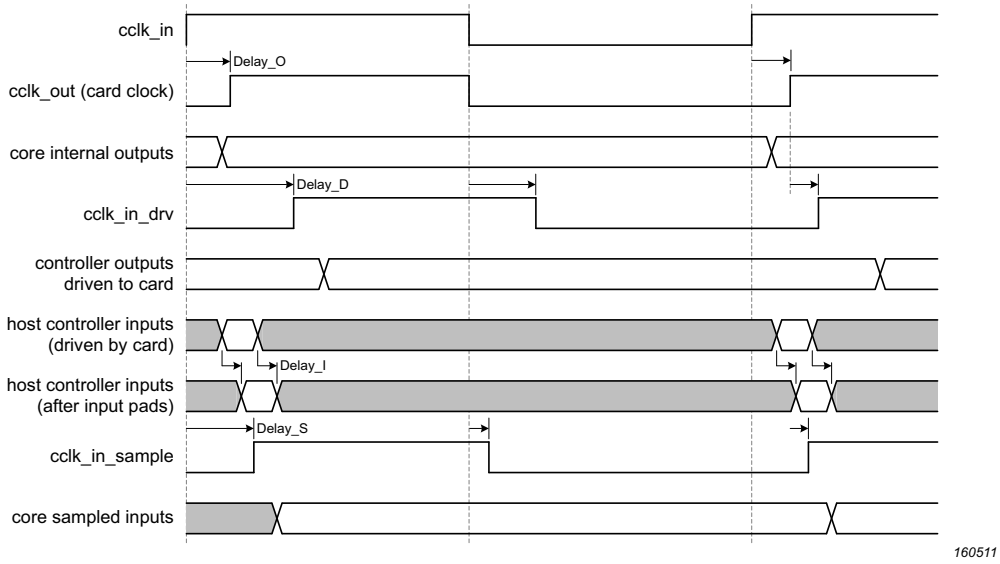


Fig 86. Clock relationship

23.9.2 Clock requirements and recommendations

There are several clock requirements and recommendations when interfacing the SDMMC/SDIO interface with cards. Table 516 lists timing requirements for the various speed modes.

Table 516. Timing requirements

Speed Mode	Max cclk_out Frequency		Min Hold Time	Min Setup Time	Min tODLY	Max tODLY	
	MHz	ns				ns	UI
SDR25 (High Speed mode)	50	20	2.0	6.0	0	14.0	0.7
SDR12	25	40	5.0	5.0	-	14.0	0.35
Identification Mode	0.4	2500	5.0	5.0	-	50.0	0.02
MMC High Speed (DAT and CMD)	50	20	3.0	3.0	-	13.7	0.685

23.9.2.1 Clock generation recommendations

The following are recommendations for clock generation:

- Figure 87 The cclk_in_drv and cclk_in_sample clocks are phase-shifted versions of cclk_in. The value of the phase shift should be selectable, based on the speed modes. The phase shift can have a resolution of 90° or 45° with respect to the cclk_in clock period—the higher the resolution, the better it is. All clocks have a 50% duty cycle.

23.9.2.2 Clock phase-shift technique

illustrates a technique for achieving phase shifts on the clocks. The phase shifter provides phase shifts of 0°, 90°, 180°, and 270°.

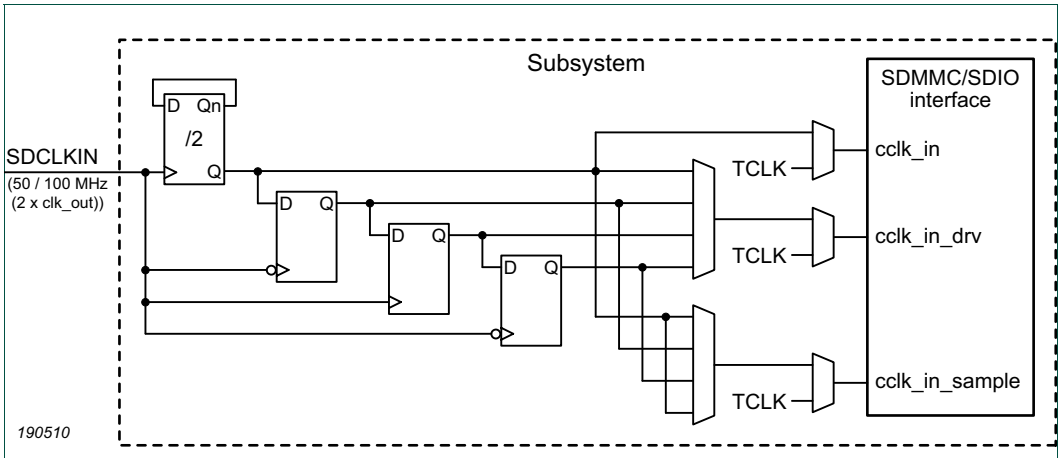


Fig 87. Clock phase-shift technique

shows the timing diagram for phase-shifted clocks.

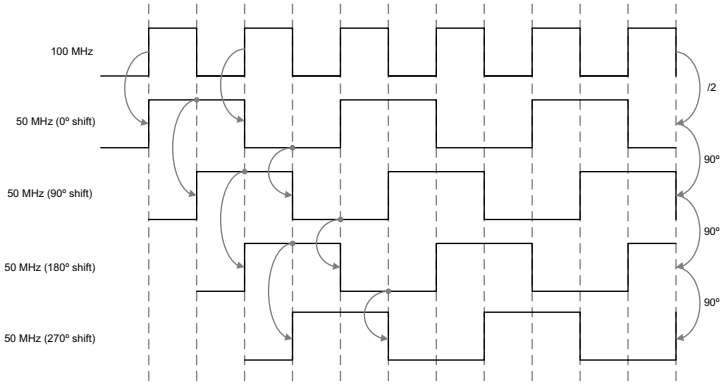


Fig 88. Timing diagram for phase-shifted clocks

23.9.2.3 SDIOCLKCTRL register

The SDIOCLKCTRL is a 32-bit SYSCON register that allows delay the SD/MMC internal input clock for both sampling of input data from the SD card and delay of data output (drive) from the LPC55S6x/LPC55S2x/LPC552x to the SD card. Sampling and drive delays are unique settings. See Section 4.5.68 “SDIO CCLKIN phase and delay control” for a detailed description of SDIOCLKCTRL.

Sample delay shifts the clocking of input data from the SD card into the SDIO interface.

The delays can be programmed in two ways.

1. The programmable delay is multiples of 250ps up to 31*250ps = 7.75ns. There are separate programmable delay settings for sample and drive delays each with a required enable bit.
2. Delay by phase shifts of 0, 90, 180, 270 of a 2X input clock. Setting the phase active bit means the input clock is 2X the actual SD output clock. If running the interface at 52MHz (26MB/s with a 4-bit interface), use a 104MHz input clock. 90 degree translates to 0.25/104e6 = 2.4ns delay. There are separate phase shift settings for sample and drive delays.

   If either the programmable delay or phase shifts are enabled, SD commands must be sent with CMD register bit 29, USE_HOLD_REG set.

   Programmable delays and phase shifts are additive when both are employed. If the input clock is 100MHz, programmable delay and phase shift is employed for the sample clock, and programmable delay is set to 10 and phase shift is set to 180 degrees, then the total delay is (10*250e-12) + (0.50/100e6) = 2.5ns + 5ns = 7.5ns.

23.9.2.4 Stop clock

Alternatively, you can avoid a stop-clock scenario by correctly enabling the Card Read Threshold feature and programming the Card Read Threshold Size—RX_WMARK and MSIZE; for details, refer to *Card Read Threshold Programming Sequence* on page 227.

For this method, it is recommended that the minimum FIFO size should be equal to the largest block size of a transfer that can be supported by the SDMMC/SDIO interface. For example, if the largest block size of the transfer that can be supported is 512 bytes and the FIFO width is 32 bits, the minimum FIFO size is 128 locations. Choosing a FIFO depth that is two or more times the maximum Block Size that the SDMMC/SDIO interface requires gives a greater performance and flexibility when choosing the Burst Size (MSIZE) and Rx Threshold (RX_WMARK). This also helps achieve best throughput.

24.1 How to read this chapter

The SCTimer/PWM is available on all LPC55S6x/LPC55S2x/LPC552x devices.

Remark: For a detailed description of SCTimer/PWM applications and code examples, see Ref. 2 “AN11538”.

24.2 Features

- The SCTimer/PWM supports:
  - Eight inputs.
  - Ten outputs.
  - Sixteen match/capture registers.
  - Sixteen events.
  - Thirty two states.
- 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 ten match and capture registers total.
  - When there is a match and/or an input or output transition or level, create events to accomplish any or all of the following: stop, limit or halt the timer; change counting direction; set, clear or toggle outputs; change the state; capture the counter value; generate an interrupt or DMA request.
  - 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.
  - PWM behavior can change based on the current state to create very complex, variable waveforms. In effect, states are a means of context switching for the entire SCT.
  - Up to ten single-edge or eight 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 a rising or falling edge or level, a combination of match and/or input/output condition. Event creation is qualified by states *(contexts*).
  - In bidirectional mode, events can be enabled based on the count direction.

- Selected events can limit, halt, start, or stop a counter or change its direction.
- Events trigger state changes, output transitions, timer captures, interrupts, and DMA transactions.
- Match register 0 can be used as an automatic limit.
- Matches can be defined as “greater/less-than-or-equal-to” the counter value for purposes of event generation.

- State control features:
  - States have no pre-defined meaning. Entirely determined by the user. States provide a mechanism for context switching for the SCT including creation of complex state machines.
  - The only function a state serves is to define which events can occur in that state.
  - A state changes to some other state in response to an event.
  - Each event can be enabled to occur in one or more states.
  - State variable allows sequencing across multiple counter cycles.

24.3 Basic configuration

Configure the SCT as follows:

- Enable the clock to the SCTimer/PWM (SCT) in the AHBCLKCTRL1 register, see Section 4.5.18 “AHB clock control 1” to enable the register interface and the peripheral clock.
- Clear the SCT peripheral reset using the PRESETCTRL register, see Section 4.5.8 “Peripheral reset control 1”.
- The SCT provides an interrupt to the NVIC, see Chapter 3 “LPC55S6x/LPC55S2x/LPC552x Nested Vectored Interrupt Controller (NVIC)”.
- SCT inputs are selected from the SCT input multiplexer registers. See Chapter 18 “LPC55S6x/LPC55S2x/LPC552x Input Multiplexing (INPUTMUX)”.
- The SCT DMA request lines are connected to the DMA trigger inputs via the DMA_ITRIG_PINMUX registers. See Section 18.5.4 “DMA trigger input multiplexing”.

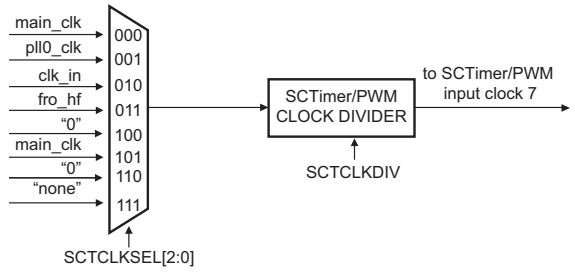


Fig 89. SCTimer/PWM clocking

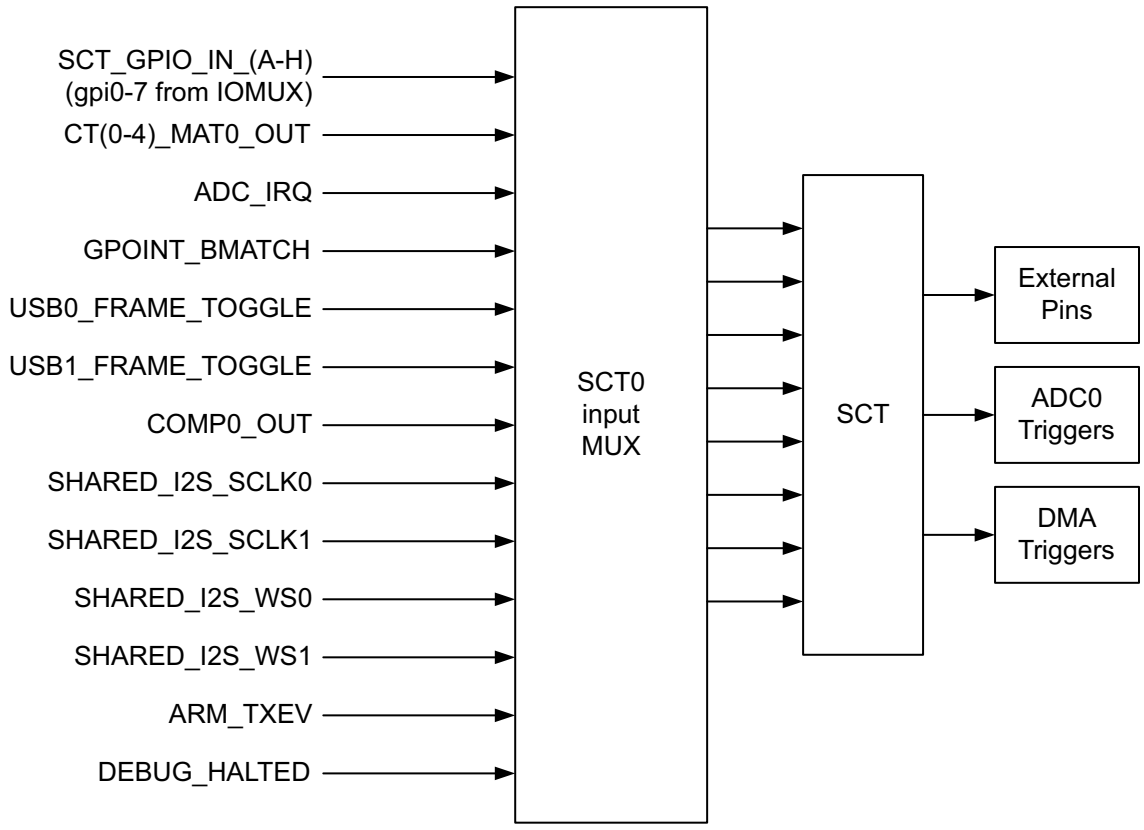


Fig 90. SCTimer/PWM connections

24.4 Pin description

Remark: Availability of inputs or outputs related to a particular peripheral function might be package dependent.

See Chapter 15 “LPC55S6x/LPC55S2x/LPC552x I/O Pin Configuration (IOCON)” to assign the SCT functions to external pins.

SCT inputs are selected from the SCT input multiplexer registers. See Chapter 18 “LPC55S6x/LPC55S2x/LPC552x Input Multiplexing (INPUTMUX)”.

SCT outputs can be routed to multiple places and can be connected to both a pin and an ADC trigger at the same time. See Chapter 18 “LPC55S6x/LPC55S2x/LPC552x Input Multiplexing (INPUTMUX)”.

Table 517. SCT0 pin description (internal signals)

Type	Connect to	Reference
Internal signals	ADC0_THCMP_IRQ, CTIMER0_MAT0, CTIMER1_MAT0, CTIMER2_MAT0, CTIMER3_MAT0, CTIMER4_MAT0, PINT_BMATCH, USB0_FRAME_TOGGLE, USB1_FRAME_TOGGLE, COMP0_OUT, SHARED_I2S_SCLK0, SHARED_I2S_SCLK1, SHARED_I2S_WS0, SHARED_I2S_WS1, ARM_TXEV, DEBUG_HALTED	Figure 90

Table 518. SCT0 pin description (inputs)

Type	Function	Connect to	Use register	Reference
External from pin	SCT0_GPI0	PIO0_0, PIO0_13, PIO0_24, PIO1_5	IOCON register for the related pin	See Chapter 15
	SCT0_GPI1	PIO0_1, PIO0_14, PIO0_25		
	SCT0_GPI2	PIO0_2, PIO0_20		
	SCT0_GPI3	PIO0_3, PIO0_21, PIO1_6		
	SCT0_GPI4	PIO0_4, PIO1_0, PIO1_7		
	SCT0_GPI5	PIO0_5, PIO1_1, PIO1_22		
	SCT0_GPI6	PIO0_6, PIO1_2, PIO1_29		
	SCT0_GPI7	PIO0_12, PIO0_17, PIO1_19, PIO1_30		
Internal	-	ADC0 trigger	SCT0 output 4, SCT0 output 5, SCT0 output 6	Table 753
Internal	-	SDMA trigger	SCT_DMA0, SCT_DMA1	Table 428

Table 519. SCT0 pin description (outputs)

Type	Function	Connect to	Use register	Reference
External to pin	SCT0_OUT0	PIO0_2, PIO0_17, PIO1_4, PIO1_23	IOCON register for the related pin	Chapter 15
	SCT0_OUT1	PIO0_3, PIO0_18, PIO1_8, PIO1_24		
	SCT0_OUT2	PIO0_10, PIO0_15, PIO0_19, PIO1_9, PIO1_25		
	SCT0_OUT3	PIO0_22, PIO0_31, PIO1_10, PIO1_26		
	SCT0_OUT4	PIO0_23, PIO1_3, PIO1_17		
	SCT0_OUT5	PIO0_26, PIO1_18		
	SCT0_OUT6	PIO0_27, PIO1_31		
	SCT0_OUT7	PIO0_28, PIO1_19		
	SCT0_OUT8	PIO0_29		
	SCT0_OUT9	PIO0_30		
Internal	-	ADC0 trigger	SCT0 output 4, SCT0 output 5, SCT0 output 6	Table 753
Internal	-	SDMA trigger	SCT_DMA0, SCT_DMA1	Table 428

Table 520. Suggested SCT input pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
11	Reserved	Reserved	Not used, set to 0.
10	Reserved	Not used, set to 0.	Not used, set to 0.
9	OD: Set to 0	Same as type D.	Not used, set to 0.
8	DIGIMODE: Set to 1.	Same as type D.	Same as type D.
7	INVERT: Set to 0.	Same as type D.	Same as type D.
6	Not used, set to 0.	Same as type D.	Not used, set to 0.

Table 520. Suggested SCT input pin settings *…continued*

IOCON bit(s)	Type D pin	Type A pin	Type I pin
5:4	MODE: Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).	Same as type D.	Not used, set to 00.
3:0	FUNC: Not used, set to 0. Specific pin inputs are directly connected to the SCT.	Same as type D.	Same as type D.
General comment	A good choice for an SCT input.	A reasonable choice for an SCT input.	A reasonable choice for an SCT input.

Recommended IOCON settings are shown in Table 520 and Table 521.

Table 521. Suggested SCT output pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
11	Reserved	Reserved	Not used, set to 0.
10	Reserved	Reserved	Reserved
9	OD: Set to 0	Reserved	Same as type D.
8	DIGIMODE: Set to 1.	Same as type D.	Same as type D.
7	INVERT: Set to 0.	Same as type D.	Same as type D.
6	Not used, set to 0.	Same as type D.	I2CSLEW: Set to 1.
5:4	MODE: Set to 0.	Same as type D.	Not used, set to 0.
3:0	FUNC: Must select the correct function for this peripheral.	Same as type D.	Same as type D.
General comment	A good choice for an SCT output.	A reasonable choice for an SCT output.	Not recommended for SCT outputs.

24.5 General description

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

The SCT can operate as a single 32-bit counter or as two independent, 16-bit counters in Unidirectional or Bidirectional mode. Software access to 16-bit registers when configured as two 16-bit counters, some limitations should be noted, see Section 24.5.1 “Important notes on using the SCT as two 16-bit counters”.

As with most timers, the SCT supports a selection of match registers against which the count value can be compared, and capture registers where the current count value can be recorded when some pre-defined condition is detected.

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

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

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

The SCT allows the user to group and filter events, thereby selecting some events to be enabled together while others are disabled in a given context. A group of enabled and disabled events can be described as a state (or a *context*), and multiple states with different sets of enabled and disabled events are allowed. Changing from one state to another is event driven as well and can therefore happen without software intervention. Any event can dictate whether to remain in the current state or switch to a new one. By defining these states, the SCTimer/PWM provides the means to periodically alter the entire behavior of the machine based on whatever criteria the user chooses. It is also possible to generate finite state machines in hardware with any desired level of complexity to accomplish complex waveform and timing tasks.

In a simple system, such as a basic timer/counter with capture and match capabilities, there is no need to use more than a single state. All events that could cause the timer to capture the timer value or toggle a match output are enabled at all times while the counter is running. In this case, no events are filtered and the system is described by a single state that does not change. it is the default configuration of the SCT.

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

For even more advanced applications, up to 32 different states/contexts can be defined (depending on the number of states available on a particular part). If required, the use of states can permit the SCTimer/PWM to serve as finite state machine generator. The ability to perform switching between groups of events provides the SCT the unique capability to be utilized as a highly complex state machine engine. Events identify the occurrence of conditions that warrant state changes and determine the next state to move to. It provides an extremely powerful control tool - particularly when the SCT inputs and outputs are connected to other on-chip resources (such as ADC triggers, other timers etc.) in addition to general-purpose I/O.

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

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

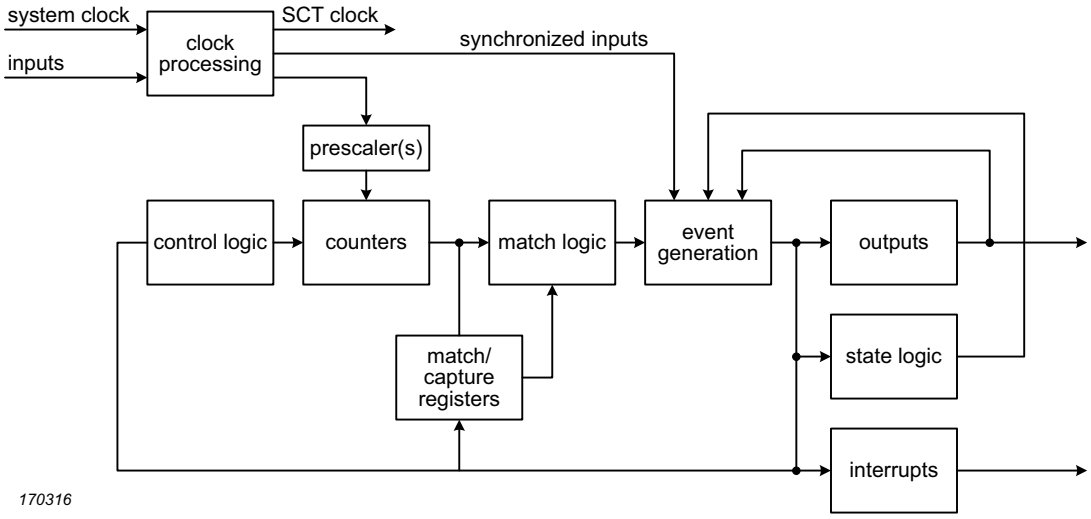


Fig 91. SCTimer/PWM block diagram

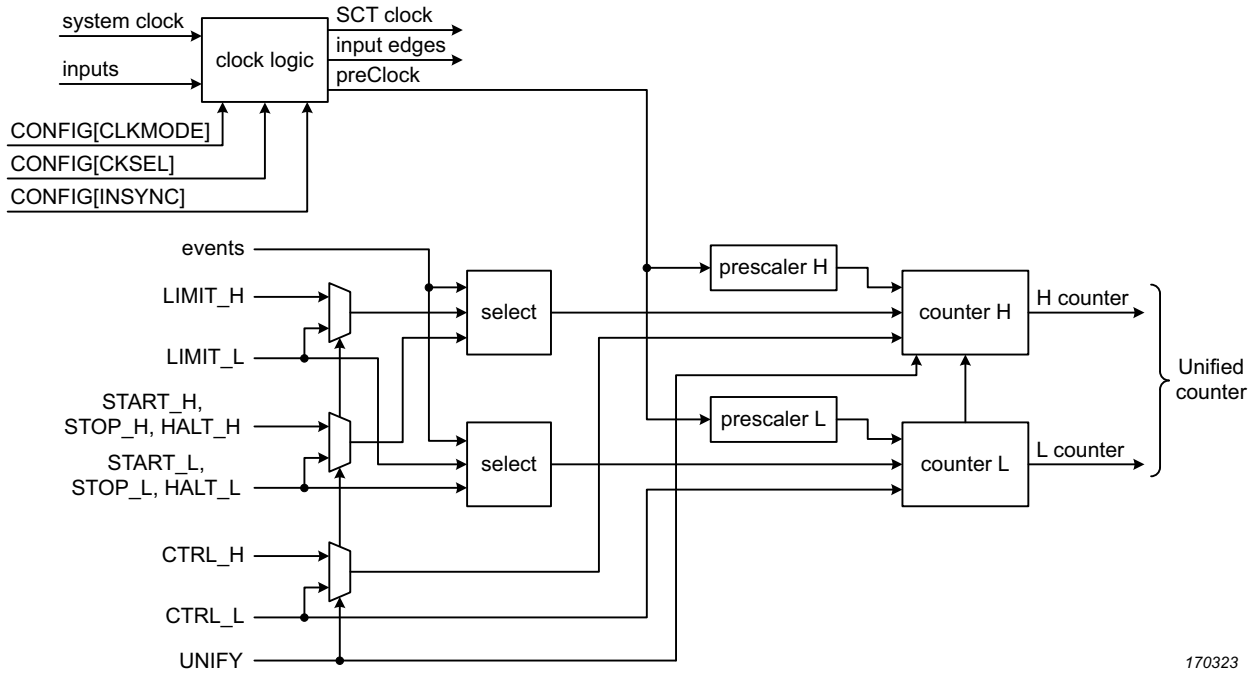


Fig 92. SCTimer/PWM counter and select logic

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

24.5.1 Important notes on using the SCT as two 16-bit counters

The implementation of the SCT on this device has a limitation that it is not possible to do half-word writes to the upper half of registers, such as (for example) CTRL_H. For the

most part, this can be dealt with in software by reading the entire word register, making changes to the upper half-word, and writing back the entire value. Note that for registers CTRL_H, STATE_H, and MATCH_H, this can only be done if both halves of the timer are in HALT mode (halted). Also see the BUSERRH flag in the CONFLAG register.

24.6 Register description

The register addresses of the SCTimer/PWM are shown in Table 522. 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 individually (for operation as two 16-bit counter/timers). The L registers can also be written individually, but the H register must be written as a word along with the L register.

   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 = 0: Registers operate as match and reload registers.
   - REGMODEn = 1: Registers operate as capture and capture control registers.

Table 522. Register overview: SCTimer/PWM (base address = 0x4008 5000)

Name	Access	Offset	Description	Reset value	Section
CONFIG	R/W	0x000	SCT configuration register.	0x0001 FE00	24.6.2
CTRL	R/W	0x004	SCT control register.	0x0004 0004	24.6.3
CTRL_L	R/W	0x004	SCT control register low counter 16-bit.	0x0000 0004	24.6.3
CTRL_H	R/W	0x006	SCT control register high counter 16-bit.	0x0000 0004	24.6.3
LIMIT	R/W	0x008	SCT limit event select register.	0x0000 0000	24.6.4
LIMIT_L	R/W	0x008	SCT limit event select register low counter 16-bit.	0x0000 0000	24.6.4
LIMIT_H	R/W	0x00A	SCT limit event select register high counter 16-bit.	0x0000 0000	24.6.4
HALT	R/W	0x00C	SCT halt event select register.	0x0000 0000	24.6.5
HALT_L	R/W	0x00C	SCT halt event select register low counter 16-bit.	0x0000 0000	24.6.5
HALT_H	R/W	0x00E	SCT halt event select register high counter 16-bit.	0x0000 0000	24.6.5
STOP	R/W	0x010	SCT stop event select register.	0x0000 0000	24.6.6
STOP_L	R/W	0x010	SCT stop event select register low counter 16-bit.	0x0000 0000	24.6.6
STOP_H	R/W	0x012	SCT stop event select register high counter 16-bit.	0x0000 0000	24.6.6
START	R/W	0x014	SCT start event select register.	0x0000 0000	24.6.7
START_L	R/W	0x014	SCT start event select register low counter 16-bit.	0x0000 0000	24.6.7
START_H	R/W	0x016	SCT start event select register high counter 16-bit.	0x0000 0000	24.6.7
COUNT	R/W	0x040	SCT counter register.	0x0000 0000	24.6.8
COUNT_L	R/W	0x040	SCT counter register low counter 16-bit.	0x0000 0000	24.6.8

Table 522. Register overview: SCTimer/PWM (base address = 0x4008 5000) *…continued*

Name	Access	Offset	Description	Reset value	Section
COUNT_H	R/W	0x042	SCT counter register high counter 16-bit.	0x0000 0000	24.6.8
STATE	R/W	0x044	SCT state register.	0x0000 0000	24.6.9
STATE_L	R/W	0x044	SCT state register low counter 16-bit.	0x0000 0000	24.6.9
STATE_H	R/W	0x046	SCT state register high counter 16-bit.	0x0000 0000	24.6.9
INPUT	RO	0x048	SCT input register.	0x0000 0000	24.6.10
REGMODE	R/W	0x04C	SCT match/capture mode register.	0x0000 0000	24.6.11
REGMODE_L	R/W	0x04C	SCT match/capture mode register low counter 16-bit.	0x0000 0000	24.6.11
REGMODE_H	R/W	0x04E	SCT match/capture registers mode register high counter 16-bit.	0x0000 0000	24.6.11
OUTPUT	R/W	0x050	SCT output register.	0x0000 0000	24.6.12
OUTPUTDIRCTRL	R/W	0x054	SCT output counter direction control register.	0x0000 0000	24.6.13
RES	R/W	0x058	SCT conflict resolution register.	0x0000 0000	24.6.14
DMAREQ0	R/W	0x05C	SCT DMA request 0 register.	0x0000 0000	24.6.15
DMAREQ1	R/W	0x060	SCT DMA request 1 register.	0x0000 0000	24.6.15
EVEN	R/W	0x0F0	SCT event interrupt enable register.	0x0000 0000	24.6.16
EVFLAG	R/W	0x0F4	SCT event flag register.	0x0000 0000	24.6.17
CONEN	R/W	0x0F8	SCT conflict interrupt enable register.	0x0000 0000	24.6.18
CONFLAG	R/W	0x0FC	SCT conflict flag register.	0x0000 0000	24.6.19
MATCH0 to MATCH15	R/W	0x100 to 0x13C	SCT match value register of match channels 0 to 15; REGMODE0 to REGMODE15 = 0.	0x0000 0000	24.6.20
MATCH0_L to MATCH15_L	R/W	0x100 to 0x13C	SCT match value register of match channels 0 to 15; low counter 16-bit; REGMODE0_L to REGMODE15_L = 0.	0x0000 0000	24.6.20
MATCH0_H to MATCH15_H	R/W	0x102 to 0x13E	SCT match value register of match channels 0 to 15; high counter 16-bit; REGMODE0_H to REGMODE15_H = 0.	0x0000 0000	24.6.20
CAP0 to CAP15	R/W	0x100 to 0x13C	SCT capture register of capture channel 0 to 15; REGMODE0 to REGMODE15 = 1.	0x0000 0000	24.6.21
CAP0_L to CAP15_L	R/W	0x100 to 0x13C	SCT capture register of capture channel 0 to 15; low counter 16-bit; REGMODE0_L to REGMODE15_L = 1.	0x0000 0000	24.6.21
CAP0_H to CAP15_H	R/W	0x102 to 0x13E	SCT capture register of capture channel 0 to 15; high counter 16-bit; REGMODE0_H to REGMODE15_H = 1.	0x0000 0000	24.6.21
MATCHREL0 to MATCHREL15	R/W	0x200 to 0x23C	SCT match reload value register 0 to 15; REGMODE0 = 0 to REGMODE15 = 0.	0x0000 0000	24.6.22
MATCHREL0_L to MATCHREL15_L	R/W	0x200 to 0x23C	SCT match reload value register 0 to 15; low counter 16-bit; REGMODE0_L = 0 to REGMODE15_L = 0.	0x0000 0000	24.6.22
MATCHREL0_H to MATCHREL15_H	R/W	0x202 to 0x23E	SCT match reload value register 0 to 15; high counter 16-bit; REGMODE0_H = 0 to REGMODE15_H = 0.	0x0000 0000	24.6.22
CAPCTRL0 to CAPCTRL15	R/W	0x200 to 0x23C	SCT capture control register 0 to 15; REGMODE0 = 1 to REGMODE15 = 1.	0x0000 0000	24.6.23
CAPCTRL0_L to CAPCTRL15_L	R/W	0x200 to 0x23C	SCT capture control register 0 to 15; low counter 16-bit; REGMODE0_L = 1 to REGMODE15_L = 1.	0x0000 0000	24.6.23
CAPCTRL0_H to CAPCTRL15_H	R/W	0x202 to 0x23E	SCT capture control register 0 to 15; high counter 16-bit; REGMODE0 = 1 to REGMODE15 = 1.	0x0000 0000	24.6.23
EV0_STATE	R/W	0x300	SCT event state register 0.	0x0000 0000	24.6.24
EV0_CTRL	R/W	0x304	SCT event control register 0.	0x0000 0000	24.6.25

Table 522. Register overview: SCTimer/PWM (base address = 0x4008 5000) *…continued*

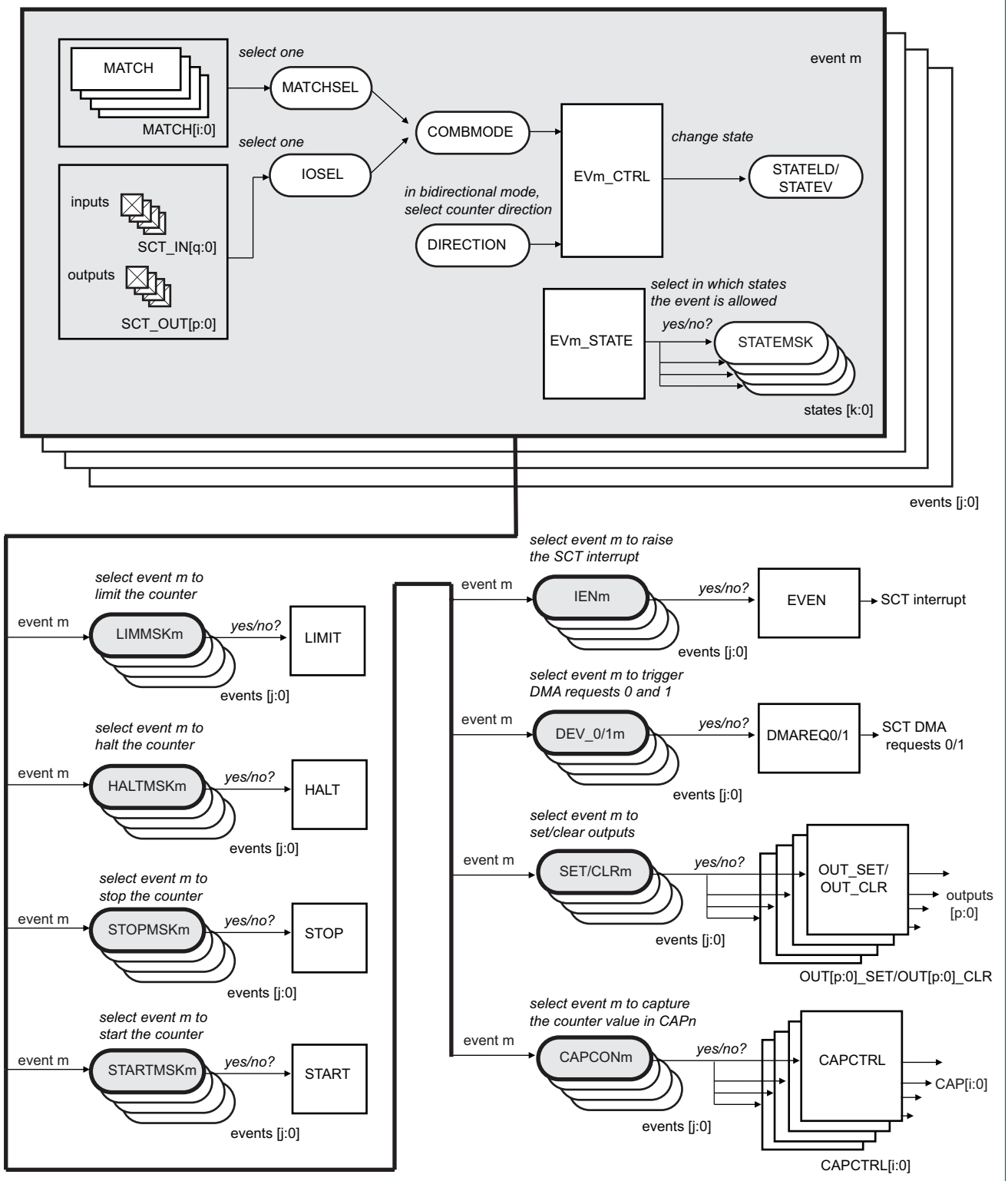
Name	Access	Offset	Description	Reset value	Section
EV1_STATE	R/W	0x308	SCT event state register 1.	0x0000 0000	24.6.24
EV1_CTRL	R/W	0x30C	SCT event control register 1.	0x0000 0000	24.6.25
EV2_STATE	R/W	0x310	SCT event state register 2.	0x0000 0000	24.6.24
EV2_CTRL	R/W	0x314	SCT event control register 2.	0x0000 0000	24.6.25
EV3_STATE	R/W	0x318	SCT event state register 3.	0x0000 0000	24.6.24
EV3_CTRL	R/W	0x31C	SCT event control register 3.	0x0000 0000	24.6.25
EV4_STATE	R/W	0x320	SCT event state register 4.	0x0000 0000	24.6.24
EV4_CTRL	R/W	0x324	SCT event control register 4.	0x0000 0000	24.6.25
EV5_STATE	R/W	0x328	SCT event state register 5.	0x0000 0000	24.6.24
EV5_CTRL	R/W	0x32C	SCT event control register 5.	0x0000 0000	24.6.25
EV6_STATE	R/W	0x330	SCT event state register 6.	0x0000 0000	24.6.24
EV6_CTRL	R/W	0x334	SCT event control register 6.	0x0000 0000	24.6.25
EV7_STATE	R/W	0x338	SCT event state register 7.	0x0000 0000	24.6.24
EV7_CTRL	R/W	0x33C	SCT event control register 7.	0x0000 0000	24.6.25
EV8_STATE	R/W	0x340	SCT event state register 8.	0x0000 0000	24.6.24
EV8_CTRL	R/W	0x344	SCT event control register 8.	0x0000 0000	24.6.25
EV9_STATE	R/W	0x348	SCT event state register 9.	0x0000 0000	24.6.24
EV9_CTRL	R/W	0x34C	SCT event control register 9.	0x0000 0000	24.6.25
EV10_STATE	R/W	0x350	SCT event state register 10.	0x0000 0000	24.6.24
EV10_CTRL	R/W	0x354	SCT event control register 10.	0x0000 0000	24.6.25
EV11_STATE	R/W	0x358	SCT event state register 11.	0x0000 0000	24.6.24
EV11_CTRL	R/W	0x35C	SCT event control register 11.	0x0000 0000	24.6.25
EV12_STATE	R/W	0x360	SCT event state register 12.	0x0000 0000	24.6.24
EV12_CTRL	R/W	0x364	SCT event control register 12.	0x0000 0000	24.6.25
EV13_STATE	R/W	0x368	SCT event state register 13.	0x0000 0000	24.6.24
EV13_CTRL	R/W	0x36C	SCT event control register 13.	0x0000 0000	24.6.25
EV14_STATE	R/W	0x370	SCT event state register 14.	0x0000 0000	24.6.24
EV14_CTRL	R/W	0x374	SCT event control register 14.	0x0000 0000	24.6.25
EV15_STATE	R/W	0x378	SCT event state register 15.	0x0000 0000	24.6.24
EV15_CTRL	R/W	0x37C	SCT event control register 15.	0x0000 0000	24.6.25
OUT0_SET	R/W	0x500	SCT output 0 set register.	0x0000 0000	24.6.26
OUT0_CLR	R/W	0x504	SCT output 0 clear register.	0x0000 0000	24.6.27
OUT1_SET	R/W	0x508	SCT output 1 set register.	0x0000 0000	24.6.26
OUT1_CLR	R/W	0x50C	SCT output 1 clear register.	0x0000 0000	24.6.27
OUT2_SET	R/W	0x510	SCT output 2 set register.	0x0000 0000	24.6.26
OUT2_CLR	R/W	0x514	SCT output 2 clear register.	0x0000 0000	24.6.27
OUT3_SET	R/W	0x518	SCT output 3 set register.	0x0000 0000	24.6.26
OUT3_CLR	R/W	0x51C	SCT output 3 clear register.	0x0000 0000	24.6.27
OUT4_SET	R/W	0x520	SCT output 4 set register.	0x0000 0000	24.6.26
OUT4_CLR	R/W	0x524	SCT output 4 clear register.	0x0000 0000	24.6.27
OUT5_SET	R/W	0x528	SCT output 5 set register.	0x0000 0000	24.6.26

Table 522. Register overview: SCTimer/PWM (base address = 0x4008 5000) *…continued*

Name	Access	Offset	Description	Reset value	Section
OUT5_CLR	R/W	0x52C	SCT output 5 clear register.	0x0000 0000	24.6.27
OUT6_SET	R/W	0x530	SCT output 6 set register.	0x0000 0000	24.6.26
OUT6_CLR	R/W	0x534	SCT output 6 clear register.	0x0000 0000	24.6.27
OUT7_SET	R/W	0x538	SCT output 7 set register.	0x0000 0000	24.6.26
OUT7_CLR	R/W	0x53C	SCT output 7 clear register.	0x0000 0000	24.6.27
OUT8_SET	R/W	0x540	SCT output 8 set register.	0x0000 0000	24.6.26
OUT8_CLR	R/W	0x544	SCT output 8 clear register.	0x0000 0000	24.6.27
OUT9_SET	R/W	0x548	SCT output 9 set register.	0x0000 0000	24.6.26
OUT9_CLR	R/W	0x54C	SCT output 9 clear register.	0x0000 0000	24.6.27

24.6.1 Register functional grouping

Most SCT registers either configure an event or select an event for a specific action of the counter (or counters) and outputs. Figure 93 shows the registers and register bits that need to be configured for each event.



Note: In this figure, letters are used to represent the maximum quantity of certain SCTimer/PWM features as noted below.

i = match/captures, j = events, k = states, p = outputs, q = inputs

Fig 93. SCT event configuration and selection registers

24.6.1.1 Counter configuration and control registers

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

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

24.6.1.2 Event configuration registers

Each event is associated with two registers:

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

24.6.1.3 Match and capture registers

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

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

24.6.1.4 Event select registers for the counter operations

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

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

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

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

24.6.1.5 Event select registers for setting or clearing the outputs

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

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

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

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

24.6.1.6 Event select registers for capturing a counter value

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

24.6.1.7 Event select register for initiating DMA transfers

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

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

24.6.1.8 Interrupt handling registers

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

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

24.6.1.9 Registers for controlling SCT inputs and outputs by software

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

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

24.6.2 SCT configuration register

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

Table 523. SCT configuration register (CONFIG, offset = 0x000)

Bit	Symbol	Value	Description	Reset value
0	UNIFY		SCT operation.	0
		0	The SCT operates as two 16-bit counters named COUNTER_L and COUNTER_H.	
		1	The SCT operates as a unified 32-bit counter.	
2:1	CLKMODE		SCT clock mode.	0
		0x0	System clock mode. The system clock clocks the entire SCT module including the counter(s) and counter prescalers.	
		0x1	Sampled system clock mode. The system clock clocks the SCT module, but the counter and prescalers are only enabled to count when the designated edge is detected on the input selected by the CKSEL field. The minimum pulse width on the selected clock-gate input is 1 bus clock period. This mode is the high-performance, sampled-clock mode.	
		0x2	SCT input clock mode. The input/edge selected by the CKSEL field clocks the SCT module, including the counters and prescalers, after first being synchronized to the system clock. The minimum width of the positive and negative phases of the clock input must each be greater than one full period of the bus/system clock.	
		0x3	Asynchronous mode. The entire SCT module is clocked directly by the input/edge selected by the CKSEL field. In this mode, the SCT outputs are switched synchronously to the SCT input clock and not the system clock. The input clock rate must be at least half the system clock rate and can be the same or faster than the system clock.	
6:3	CKSEL		SCT clock select. The specific functionality of the designated input/edge is dependent on the CLKMODE bit selection in this register.	0
		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.	
		0x8	Rising edges on input 4.	
		0x9	Falling edges on input 4.	
		0xA	Rising edges on input 5.	
		0xB	Falling edges on input 5.	
		0xC	Rising edges on input 6.	
		0xD	Falling edges on input 6	
		0xE	Rising edges on input 7.	
		0xF	Falling edges on input 7.	

Table 523. SCT configuration register (CONFIG, offset = 0x000) *…continued*

Bit	Symbol	Value	Description	Reset value
7	NORELOAD_L	-	A 1 in this bit prevents the lower match registers from being reloaded from their respective reload registers. Setting this bit eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear this bit at any time. This bit applies to both the higher and lower registers when the UNIFY bit is set.	0
8	NORELOAD_H	-	A 1 in this bit prevents the higher match registers from being reloaded from their respective reload registers. Setting this bit eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear this bit at any time. This bit is not used when the UNIFY bit is set.	0
16:9	INSYNC	-	Synchronization for input N (bit 9 = input 0, bit 10 = input 1,..., bit 16 = input 7). A 1 in one of these bits subjects the corresponding input to synchronization to the SCT clock, before it is used to create an event. This synchronization injects a two SCT-clock delay in the input path. Clearing this bit bypasses synchronization on the corresponding input. This bit may be cleared for faster input response time if both of the following conditions are met (for all Clock modes): The corresponding input is already synchronous to the SCT clock. The SCT clock frequency does not exceed 100 MHz. Note: The SCT clock is the bus/system clock for CKMODE 0-2 or the selected, asynchronous input clock for CKMODE3. Alternatively, for CKMODE2 only, it is also allowable to bypass synchronization if both of the following conditions are met: The corresponding input is synchronous to the designated CKMODE2 input clock. The CKMODE2 input clock frequency is less than one-third the frequency of the bus/system clock.	0
17	AUTOLIMIT_L	-	This bit applies to the lower registers when the UNIFY bit = 0, and both the higher and lower registers when the UNIFY bit is set. Software can write to set or clear this bit at any time. 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 Unidirectional mode or to change the direction of count in Bidirectional mode.	0
18	AUTOLIMIT_H	-	This bit applies to the upper registers when the UNIFY bit = 0, and is not used when the UNIFY bit is set. Software can write to set or clear this bit at any time. 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 Unidirectional mode or to change the direction of count in Bidirectional mode.	0
31:19	-	-	Reserved.	-

24.6.3 SCT control register

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

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

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. it is true regardless of what triggered the event.

Table 524. SCT control register (CTRL, offset = 0x004)

Bit	Symbol	Value	Description	Reset value
0	DOWN_L	-	This read-only bit is 1 when the L or unified counter is counting down. Hardware sets this bit when the counter is counting up, counter limit occurs, and BIDIR = 1. Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0.	0
1	STOP_L	-	When this bit is 1 and HALT is 0, the L or unified counter does not run, but I/O events related to the counter can occur. If a designated start event occurs, this bit is cleared and counting resumes.	0
2	HALT_L	-	When this bit is 1, the L or unified counter does not run and no events can occur. A reset sets this bit. When the HALT_L bit is one, the STOP_L bit is cleared. It is possible to remove the halt condition while keeping the SCT in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. **Remark:** Once set, only software can clear this bit to restore counter operation. This bit is set on reset.	1
3	CLRCTR_L	-	When the counter is halted (not just stopped), writing a 1 to this bit will clear the L or unified counter. This bit always reads as 0.	0
4	BIDIR_L		L or unified counter direction select	0
		0	Up. The counter counts up to a limit condition, then is cleared to zero.	
		1	Up-down. The counter counts up to a limit, then counts down to a limit condition or to 0.	
12:5	PRE_L	-	Specifies the factor by which the SCT clock is prescaled to produce the L or unified counter clock. The counter clock is clocked at the rate of the SCT clock divided by PRE_L+1. **Remark:** Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.	0
15:13	-	-	Reserved.	-
16	DOWN_H	-	This read-only bit is 1 when the H counter is counting down. Hardware sets this bit when the counter is counting, a counter limit condition occurs, and BIDIR is 1. Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0.	0

Table 524. SCT control register (CTRL, offset = 0x004) *…continued*

Bit	Symbol	Value	Description	Reset value
17	STOP_H	-	When this bit is 1 and HALT is 0, the H counter does not, run but I/O events related to the counter can occur. If such an event matches the mask in the Start register, this bit is cleared and counting resumes.	0
18	HALT_H	-	When this bit is 1, the H counter does not run and no events can occur. A reset sets this bit. When the HALT_H bit is one, the STOP_H bit is cleared. It is possible to remove the halt condition while keeping the SCT in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. **Remark:** Once set, this bit can only be cleared by software to restore counter operation. This bit is set on reset.	1
19	CLRCTR_H	-	When the counter is halted (not just stopped), writing a 1 to this bit will clear 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.	-

24.6.4 SCT limit event select register

The running counter can be limited by an event. When any of the events selected in this register occur, the counter is cleared to zero from its current value or changes counting direction if in Bidirectional mode.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit causes its associated event to serve as a LIMIT event. When any limit event occurs, the counter is reset to zero in Unidirectional mode or changes its direction of count in Bidirectional mode and keeps running.To define the actual limiting event (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: Counting up to all ones or counting down to zero is always equivalent to a limit event occurring.

Note that in addition to using this register to specify events that serve as limits, it is also possible to automatically cause a limit condition whenever a match register 0 match occurs. This eliminates the need to define an event for the sole purpose of creating a limit. The AUTOLIMITL and AUTOLIMITH bits in the configuration register enable/disable this feature, see Table 523.

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 individually or in a single 32-bit read or write operation. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Table 525. SCT limit event select register (LIMIT, offset = 0x008)

Bit	Symbol	Description	Reset value
15:0	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, …). The number of bits = number of events supported by this SCT.	0
31:16	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, …). The number of bits = number of events supported by this SCT.	0

24.6.5 SCT halt event select register

The running counter can be disabled (halted) by an event. When any of the events selected in this register occur, the counter stops running and all further events are disabled.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a HALT event. To define the actual events that cause the counter to halt (a match, an I/O pin toggle, etc.), see the EVn_CTRL registers.

Remark: A HALT condition can only be removed when software clears the HALT bit in the CTRL register, seeTable 524.

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 individually or in a single 32-bit read or write operation. The L registers can also be written individually, but the H register must be written as a word along with the L register

Table 526. SCT halt event select register (HALT, offset = 0x00C)

Bit	Symbol	Description	Reset value
15:0	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, …). The number of bits = number of events supported by this SCT.	0
31:16	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, …). The number of bits = number of events supported by this SCT.	0

24.6.6 SCT stop event select register

The running counter can be stopped by an event. When any of the events selected in this register occur, counting is suspended, that is the counter stops running and remains at its current value. Event generation remains enabled, and any event selected in the START register such as an I/O event or an event generated by the other counter can restart the counter.

This register specifies which events stop the counter. Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a STOP event. To define the actual event that causes the counter to stop (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: Software can stop and restart the counter by writing to the CTRL register.

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

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

Table 527. SCT stop event select register (STOP, offset = 0x010)

Bit	Symbol	Description	Reset value
15:0	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, …). The number of bits = number of events supported by this SCT.	0
31:16	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, …). The number of bits = number of events supported by this SCT.	0

24.6.7 SCT start event select register

The stopped counter can be re-started by an event. When any of the events selected in this register occur, counting is restarted from the current counter value.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a START event. When any START event occurs, hardware will clear the STOP bit in the control register CTRL. Note that a START event has no effect on the HALT bit. Only software can remove a HALT condition. To define the actual event that starts the counter (an I/O pin toggle or an event generated by the other running counter in dual-counter mode), see the EVn_CTRL register.

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

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

Table 528. SCT start event select register (START, offset = 0x014)

Bit	Symbol	Description	Reset value
15:0	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, …). The number of bits = number of events supported by this SCT.	0
31:16	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, …). The number of bits = number of events supported by this SCT.	0

24.6.8 SCT counter register

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

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

Writing to the COUNT_L, COUNT_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). Attempting to write to the counter when it is not halted causes a bus error. Software can read the counter registers at any time.

Table 529. SCT counter register (COUNT, offset = 0x040)

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

24.6.9 SCT state register

Each group of enabled and disabled events is assigned a number called the state variable. For example, a state variable with a value of 0 could have events 0, 2, and 3 enabled and all other events disabled. A state variable with the value of 1 could have events 1, 4, and 5 enabled and all others disabled.

Remark: The EVm_STATE registers define which event is enabled in each group.

Software can read the state associated with a counter at any time. Writing to the STATE_L or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). STATE_H can only be written as a word along with STATE_L, and both counters must be halted.

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

- Set and clear outputs.
- Limit, stop, and start the counter.
- Cause interrupts and DMA requests.
- Modify the state variable.

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

A state variable can be used to track and control multiple cycles of the associated counter in any desired operational sequence. The state variable is logically associated with a state machine diagram which represents the SCT configuration. See Section 24.6.24 “SCT event enable registers 0 to 15” and Section 24.6.25 “SCT event control registers 0 to 15” for more about the relationship between states and events.

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

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

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

Table 530. SCT state register (STATE, offset = 0x044)

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.	-

24.6.10 SCT input register

Software can read the state of the SCT inputs in this read-only register in slightly different forms.

1. The AIN bit displays the state of the input captured on each rising edge of the SCT clock. This corresponds to a nearly direct read-out of the input but can cause spurious fluctuations in case of an asynchronous input signal.
2. The SIN bit displays the form of the input as it is used for event detection. This may include additional stages of synchronization, depending on what is specified for that input in the INSYNC field in the CONFIG register:
   - If the INSYNC bit is set for the input, the input is triple-synchronized to the SCT clock resulting in a stable signal that is delayed by three SCT clock cycles.
   - If the INSYNC bit is not set, the SIN bit value is identical to the AIN bit value.

Table 531. SCT input register (INPUT, offset = 0x048)

Bit	Symbol	Description	Reset value
0	AIN0	Input 0 state. Input 0 state on the last SCT clock edge.	-
1	AIN1	Input 1 state. Input 1 state on the last SCT clock edge.	-
2	AIN2	Input 2 state. Input 2 state on the last SCT clock edge.	-
3	AIN3	Input 3 state. Input 3 state on the last SCT clock edge.	-
15:4	AIN…	Input state for the remainder of inputs implemented in this SCT.	-
16	SIN0	Input 0 state. Input 0 state following the synchronization specified by INSYNC0.	-
17	SIN1	Input 1 state. Input 1 state following the synchronization specified by INSYNC1.	-
18	SIN2	Input 2 state. Input 2 state following the synchronization specified by INSYNC2.	-
19	SIN3	Input 3 state. Input 3 state following the synchronization specified by INSYNC3.	-
31:20	SIN…	Input state for the remainder of states implemented in this SCT.	-

24.6.11 SCT match/capture mode register

If UNIFY = 1 in the CONFIG register, only the _L bits of this register are used. In this case, REGMODE_H is not used.

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

The SCT contains multiple match/capture registers. The Register mode register selects whether each register acts as a match register, see Section 24.6.20 “SCT match registers 0 to 15 (REGMODEn bit = 0)”) or as a capture register, see Section 24.6.21 “SCT capture registers 0 to 15 (REGMODEn bit = 1)”. Each match/capture register has an accompanying register which functions as a reload register when the primary register is used as a match register, see Section 24.6.22 “SCT match reload registers 0 to 15 (REGMODEn bit = 0)” or as a capture control (event select) register when the register is used as a capture register, see Section 24.6.23 “SCT capture control registers 0 to 15 (REGMODEn bit = 1)”. REGMODE_H is used only when the UNIFY bit is 0.

Table 532. SCT match/capture mode register (REGMODE, offset = 0x04C)

Bit	Symbol	Description	Reset value
15:0	REGMOD_L	Each bit controls one match/capture register (register 0 = bit 0, register 1 = bit 1, …). The number of bits = number of match/captures supported by this SCT. 0 = register operates as match register. 1 = register operates as capture register.	0
31:16	REGMOD_H	Each bit controls one match/capture register (register 0 = bit 16, register 1 = bit 17, …). The number of bits = number of match/captures supported by this SCT. 0 = register operates as match registers. 1 = register operates as capture registers.	0

24.6.12 SCT output register

Each SCT output has a corresponding bit in this register to allow software to control the output state directly or read its current state.

While the counter is running, outputs are set, cleared, or toggled only by events. However, using this register, software can write to any of the output registers when both counters are halted to control the outputs directly. Writing to the OUT register is only allowed when all counters (L-counter, H-counter, or unified counter) are halted (HALT bits are set to 1 in the CTRL register).

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

Table 533. SCT output register (OUTPUT, offset = 0x050)

Bit	Symbol	Description	Reset value
15:0	OUT	Writing a 1 to bit n forces the corresponding output HIGH. Writing a 0 forces the corresponding output LOW (output 0 = bit 0, output 1 = bit 1, …). The number of bits = number of outputs in this SCT.	0
31:16	-	Reserved	-

24.6.13 SCT Bidirectional output control register

For Bidirectional mode, this register specifies (for each output) the impact of the counting direction on the meaning of set and clear operations on the output, see Section 24.6.26 “SCT output set registers 0 to 9” and Section 24.6.27 “SCT output clear registers 0 to 9”. The purpose of this register is to facilitate the creation of center-aligned output waveforms without the need to define additional events.

Table 534. SCT Bidirectional output control register (OUTPUTDIRCTRL, offset = 0x054)

Bit	Symbol	Value	Description	Reset value
1:0	SETCLR0		Set/clear operation on output 0.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
3:2	SETCLR1		Set/clear operation on output 1.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
5:4	SETCLR2		Set/clear operation on output 2.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
7:6	SETCLR3		Set/clear operation on output 3.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
9:8	SETCLR4		Set/clear operation on output 4.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
31:10	SETCLR…		Set/clear operation controls for the remainder of outputs on this SCT. [1]	0

[1] For as many outputs as are supported by the specific SCTimer/PWM.

24.6.14 SCT conflict resolution register

The output conflict resolution register specifies what action should be taken if multiple events (or even the same event) dictate that a given output should be both set and cleared at the same time.

To enable an event to toggle an output each time the event occurs, set the bits for that event in both the OUTn_SET and OUTn_CLR registers and set the On_RES value to 0x3 in this register.

Table 535. SCT conflict resolution register (RES, offset = 0x058)

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

Table 535. SCT conflict resolution register (RES, offset = 0x058) *…continued*

Bit	Symbol	Value	Description	Reset value
3:2	O1RES		Effect of simultaneous set and clear on output 1.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR1 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR1 field).	
		0x3	Toggle output.	
5:4	O2RES		Effect of simultaneous set and clear on output 2.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR2 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output n (or set based on the SETCLR2 field).	
		0x3	Toggle output.	
7:6	O3RES		Effect of simultaneous set and clear on output 3.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR3 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR3 field).	
		0x3	Toggle output.	
9:8	O4RES		Effect of simultaneous set and clear on output 4.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR4 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR4 field).	
		0x3	Toggle output.	
31:10	O…RES		Resolution controls for the remainder of outputs on this SCT. [1]	0

[1] For as many outputs as are supported by the specific SCTimer/PWM.

24.6.15 SCT DMA request 0 and 1 registers

The SCT includes two DMA request outputs. These registers enable the DMA requests to be triggered when a particular event occurs or when counter match registers are loaded from its reload registers. The DMA request registers are word-write only. Attempting to write a half-word value to these registers result in a bus error.

Event-triggered DMA requests are particularly useful for launching DMA activity to or from other peripherals under the control of the SCT.

Table 536. SCT DMA 0 request register (DMAREQ0, offset = 0x05C)

Bit	Symbol	Description	Reset value
15:0	DEV_0	If bit n is one, event n triggers DMA request 0 (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events in this SCT.	0
29:16	-	Reserved.	-
30	DRL0	A 1 in this bit triggers DMA request 0 when it loads the MATCH_L/Unified registers from the RELOAD_L/Unified registers.	0
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.	0

Table 537. SCT DMA 1 request register (DMAREQ1, offset = 0x060)

Bit	Symbol	Description	Reset value
15:0	DEV_1	If bit n is one, event n triggers DMA request 1 (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events in this SCT.	0
29:16	-	Reserved.	-
30	DRL1	A 1 in this bit triggers DMA request 1 when it loads the match L/Unified registers from the reload L/Unified registers.	0
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.	0

24.6.16 SCT event interrupt enable register

This register enables flags to request an interrupt if the FLAGn bit in the SCT event flag register, see Section 24.6.17 “SCT event flag register” is also set.

Table 538. SCT event interrupt enable register (EVEN, offset = 0x0F0)

Bit	Symbol	Description	Reset value
15:0	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, …). The number of bits = number of events supported by this SCT.	0
31:16	-	Reserved.	-

24.6.17 SCT event flag register

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

Table 539. SCT event flag register (EVFLAG, offset = 0x0F4)

Bit	Symbol	Description	Reset value
15:0	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, …). The number of bits = number of events supported by this SCT.	0
31:16	-	Reserved.	-

24.6.18 SCT conflict interrupt enable register

This register enables the no-change conflict events specified in the SCT conflict resolution register to generate an interrupt request.

Table 540. SCT conflict interrupt enable register (CONEN, offset = 0x0F8)

Bit	Symbol	Description	Reset value
15:0	NCEN	The SCT requests an interrupt when bit n of this register and the SCT conflict flag register are both one (output 0 = bit 0, output 1 = bit 1, …). The number of bits = number of outputs supported by this SCT.	0
31:16	-	Reserved.	

24.6.19 SCT conflict flag register

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

Table 541. SCT conflict flag register (CONFLAG, offset = 0x0FC)

Bit	Symbol	Description	Reset value
15:0	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, …). The number of bits = number of outputs supported by this SCT.	0
29:16	-	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

24.6.20 SCT match registers 0 to 15 (REGMODEn bit = 0)

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

Match registers can be read at any time. Writing to the MATCH_L or unified register is only allowed when the corresponding counter(s) are halted (HALT bits are set to 1 in the CTRL register). MATCH_H can only be written as a word along with MATCH_L, and both counters must be halted. 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 24.6.4 “SCT limit event select register”, 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 542. SCT match registers 0 to 15 (MATCH[0:15], offset = 0x100 (MATCH0) to 0x13C (MATCH15)) (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

24.6.21 SCT capture registers 0 to 15 (REGMODEn bit = 1)

These registers allow software to record the counter values upon occurrence of the events selected by the corresponding capture control registers occurred.

Table 543. SCT capture registers 0 to 15 (CAP[0:15], offset = 0x100 (CAP0) to 0x13C (CAP15)) (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

24.6.22 SCT match reload registers 0 to 15 (REGMODEn bit = 0)

A match register (L, H, or unified 32-bit) is loaded from its corresponding reload register at the start of each new counter cycle, that is:

- when BIDIR = 0 and the counter is cleared to zero upon reaching its limit condition.
- when BIDIR = 1 and the counter counts down to 0.

In either case, reloading does not occur if the corresponding NORELOAD bit is set in the CFG register.

Table 544. SCT match reload registers 0 to 15 (MATCHREL[0:15], offset = 0x200 (MATCHREL0) to 0x23E (MATCHREL15)) (REGMODEn bit = 0)

Bit	Symbol	Description	Reset value
15:0	RELOADn_L	When UNIFY = 0, specifies the 16-bit value to be loaded into the MATCHn_L register. When UNIFY = 1, specifies the lower 16 bits of the 32-bit value to be loaded into the MATCHn register.	0
31:16	RELOADn_H	When UNIFY = 0, specifies the 16-bit to be loaded into the MATCHn_H register. When UNIFY = 1, specifies the upper 16 bits of the 32-bit value to be loaded into the MATCHn register.	0

24.6.23 SCT capture control registers 0 to 15 (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 individually or in a single 32-bit read or write operation. The L registers can also be written individually, but the H register must be written as a word along with the L register.

The capture registers can be loaded with the current counter value when any of the specified events occur.

Each capture control register (L, H, or unified 32-bit) controls which events cause the load of corresponding capture register from the counter.

Table 545. SCT capture control registers 0 to 15(CAPCTRL[0:15], offset = 0x200 (CAPCTRL0) to 0x23C (CAPCTRL15)) (REGMODEn bit = 1)

Bit	Symbol	Description	Reset value
15:0	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, …). The number of bits = number of match/captures supported by this SCT.	0
31:16	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, …). The number of bits = number of match/captures supported by this SCT.	0

24.6.24 SCT event enable registers 0 to 15

Each event can be enabled in some contexts (or states) and disabled in others. Each event defined in the EV_CTRL register has one associated event enable register that can enable or disable the event for each available state.

An event n is completely disabled when its EVn_STATE register contains all zeros, since it is masked regardless of the current state. Unused events should be disabled in this manner.

In simple applications that do not use states, writing 0x01 (or any other value with a 1 in bit 0) will enable the event. Since the state doesn't change (that is, the state variable always remains at its reset value of 0), setting bit 0 permanently enables this event. Conversely, clearing bit 0 will disable the event.

Table 546. SCT event state mask registers 0 to 15 (EV[0:15]_STATE, offset = 0x300 (EV0_STATE) to 0x37C (EV15_STATE))

Bit	Symbol	Description	Reset value
15:0	STATEMSKn	If bit m is one, event n is enabled to occur whenever the state = m. When the UNIFY bit is 0, the pertinent state is the one associated with the counter selected by the HEVENT bit in the event control register. (n = event number, m = state number; state 0 = bit 0, state 1= bit 1, …). The number of bits = number of states in this SCT.	0
31:16	-	Reserved.	-

24.6.25 SCT event control registers 0 to 15

This register defines the conditions for an event to occur based on the counter values or input and output states.Once the event is configured, it can be selected to trigger multiple actions (for example stop the counter and toggle an output) unless the event is blocked in the current state of the SCT or the counter is halted. To block a particular event from occurring, use the EV_STATE register. To block all events for a given counter, set the HALT bit in the CTRL register or select an event to halt the counter.

An event can be programmed to occur based on a selected input or output edge or level and/or based on its counter value matching a selected match register. In bidirectional mode, events can also be enabled based on the direction of count.

When the UNIFY bit is 0, each event is associated with a particular counter by the HEVENT bit in its event control register. An event is permanently disabled when its event state mask register contains all 0s.

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

Table 547. SCT event control register 0 to 15 (EV[0:15]_CTRL, offset = 0x304 (EV0_CTRL) to 0x37C (EV15_CTRL))

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

Table 547. SCT event control register 0 to 15 (EV[0:15]_CTRL, offset = 0x304 (EV0_CTRL) to 0x37C (EV15_CTRL))

Bit	Symbol	Value	Description	Reset value
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 selected by IOSEL.	
		1	Selects the outputs selected by IOSEL.	
9:6	IOSEL	-	Selects the input or output signal number 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
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.	0
		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
		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.	0
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.	0
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.	0
		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	-

24.6.26 SCT output set registers 0 to 9

Each SCT output can be set upon the occurrence of one or more specified events.

There is one output set register for each SCT output which selects which events can set that output. Each bit of an output set register is associated with a different event (bit 0 with event 0, etc.).

Note that it is possible to reverse the action specified by *SET* and *CLR* when counting down in bidirectional mode depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the creation of the actual event(s) that sets an output (a match and an I/O pin toggle), see the EVn_CTRL register.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. It is true regardless of what triggered the event.

Table 548. SCT output set register (OUT[0:9]_SET, offset = 0x500 (OUT0_SET) to 0x548 (OUT9_SET)

Bit	Symbol	Description	Reset value
15:0	SET	A 1 in bit m selects event m to set output n (or clear it if SETCLRn = 0x1 or 0x2) output 0 = bit 0, output 1 = bit 1, … The number of bits = number of events supported by this SCT. When the counter is used in Bidirectional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTCTRL register.	0
31:16	-	Reserved.	-

24.6.27 SCT output clear registers 0 to 9

Each SCT output can be cleared upon the occurrence of one or more specified events.

There is one register for each SCT output which selects which events can clear that output. Each bit of an output clear register is associated with a different event (for example, bit 0 with event 0).

Note that it is possible to reverse the action specified by *SET* and *CLR* when counting down in Bidirectional mode depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the creation of the actual event(s) that sets an output (a match and an I/O pin toggle), see the EVn_CTRL register.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. It is true regardless of what triggered the event.

Table 549. SCT output clear register (OUT[0:9]_CLR, offset = 0x504 (OUT0_CLR) to 0x54C (OUT9_CLR))

Bit	Symbol	Description	Reset value
15:0	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, … The number of bits = number of events in this SCT. When the counter is used in Bidirectional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTCTRL register.	0
31:16	-	Reserved.	-

24.7 Functional description

24.7.1 Match logic

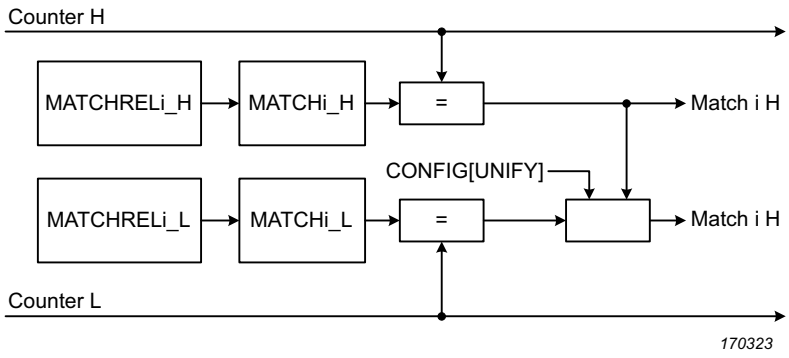


Fig 94. Match logic

24.7.2 Capture logic

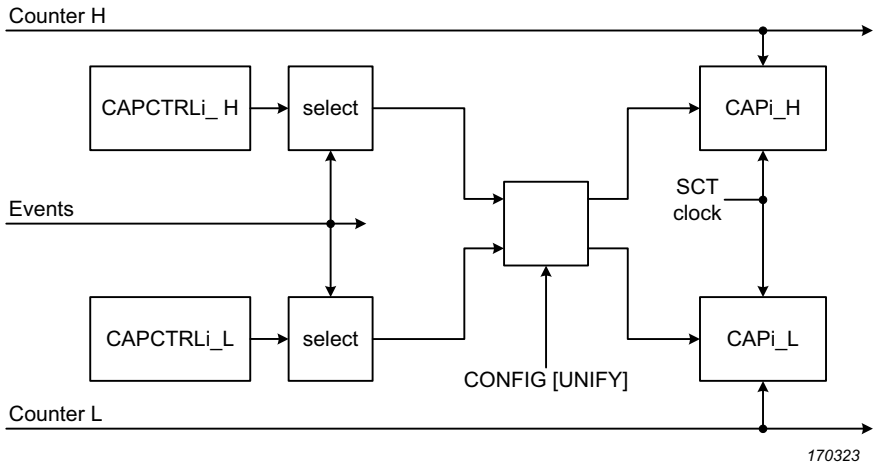


Fig 95. Capture logic

24.7.3 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.

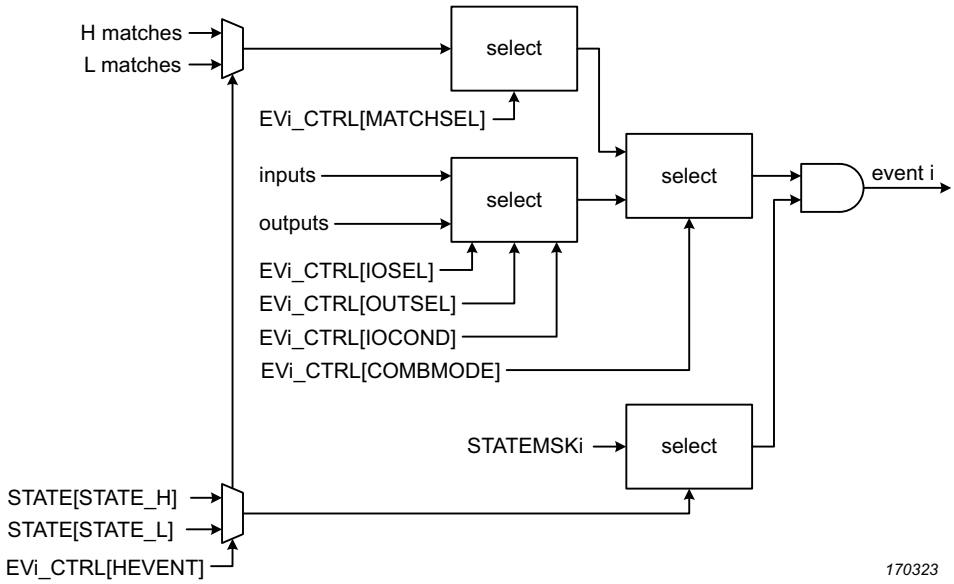


Fig 96. Event selection

24.7.4 Output generation

Figure 97 shows one output slice of the SCT.

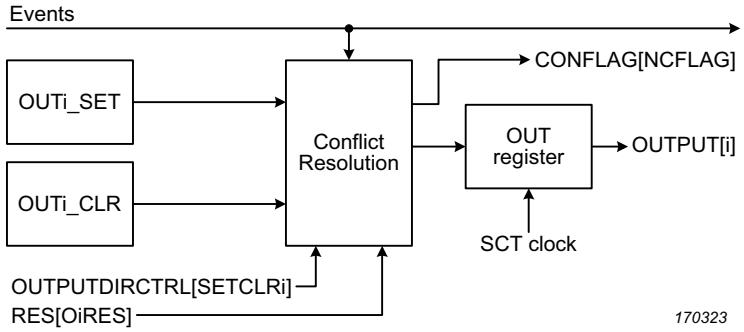


Fig 97. Output slice i

24.7.5 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 intervene 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.

24.7.6 Interrupt generation

The SCT generates one interrupt to the NVIC.

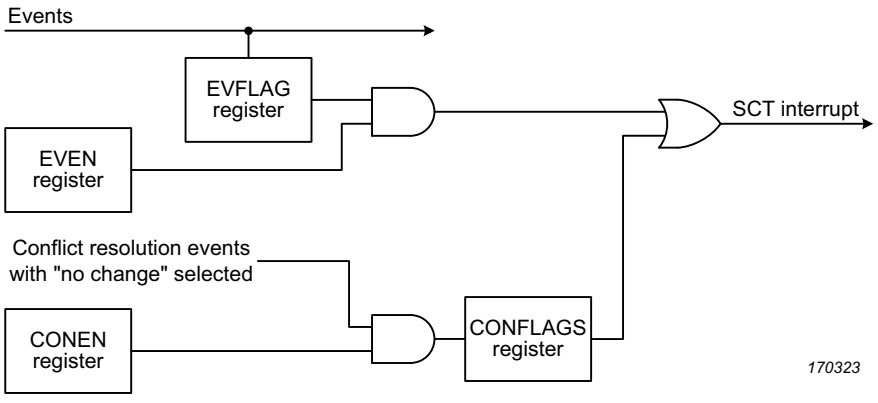


Fig 98. SCT interrupt generation

24.7.7 Clearing the pre-scaler

When enabled by a non-zero PRE field in the control register, the pre-scaler acts as a clock divider for the counter, like a fractional part of the counter value. The pre-scaler 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 pre-scaler. However, a limit event caused by a match only clears a non-zero pre-scaler in one special case as described Section 24.7.8 “Match versus I/O events”.

A limit event when BIDIR is 1 does not clear the pre-scaler. 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.

24.7.8 Match versus I/O events

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

- The pre-scaler 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 pre-scaler is enabled, and (PRELIM=0 or the pre-scaler 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 an SCT clock when its counter HALT and STOP bits are both 0 and the counter is enabled.

Table 550 shows when the various kinds of events can occur.

Table 550. 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).

24.7.9 SCT operation

In its simplest, single-state configuration, the SCT operates as an event controlled Unidirectional or Bidirectional counter. Events can be configured to occur on 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 24.7.10 “Configure the SCT”.
- To start, run, and stop the SCT, see Section 24.7.11 “Run the SCT”.
- To configure the SCT as simple event controlled counter/timer, see Section 24.7.12 “Configure the SCT without using states”.

24.7.10 Configure the SCT

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

24.7.10.1 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.

24.7.10.2 Configure the match and capture registers

1. Select how many match and capture registers the application uses (not more than what is available on this device):
   - In the REGMODE register, select for each of the 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.

24.7.10.3 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 the maximum number of outputs on this device, 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.

24.7.10.4 Configure multiple states

1. In the EVn_STATE register for each event (up to the maximum number of events on this device, 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 the maximum number of events on this device, 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.

24.7.10.5 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.

24.7.11 Run the SCT

1. Configure the SCT, see Section 24.7.10 “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 pre-scale 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.

24.7.12 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.

24.7.13 SCT PWM example

Figure 99 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 0 where the events EV0 and EV1 are enabled.

The example uses the following SCT configuration:

- One input.
- One output.
- Five match registers.
- Six events and match 0 used with autolimit function.
- Two states.

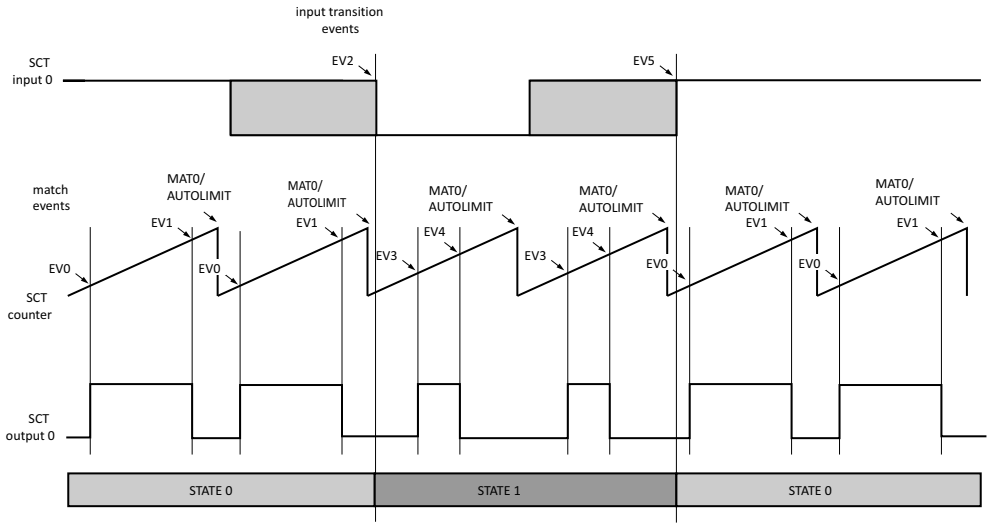


Fig 99. SCT configuration example

This application of the SCT uses the following configuration (all register values not listed in Table 551 are set to their default values):

Table 551. 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.

Table 551. SCT configuration example *…continued*

Configuration	Registers	Setting
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	MATCH 0/1/2/3/4	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	MATCHREL 0/1/2/3/4	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.
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.

Table 551. SCT configuration example *…continued*

Configuration	Registers	Setting
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.

25.1 How to read this chapter

These five standard timers are available on all LPC55S6x/LPC55S2x/LPC552x devices.

25.2 Features

- Each is a 32-bit counter/timer with a programmable 32-bit pre-scaler. The timers include external capture and match pin connections.
- Counter or timer operation.
- Each CTIMER has a selection of function clocks that may be asynchronous to other system clocks, see Section 4.5.29 through Section 4.5.33.
- Up to four 32-bit captures can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt. The number of capture inputs for each timer that are actually available on device pins may vary by device.
- The timer and pre-scaler 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.
  - Optional auto-reload from match shadow registers when counter is reset.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- For each timer, up to four external outputs corresponding to match registers with the following capabilities (the number of match outputs for each timer that are actually available on device pins may vary by device):
  - Set LOW on match.
  - Set HIGH on match.
  - Toggle on match.
  - Do nothing on match.
- Up to four match registers can be configured for PWM operation, allowing up to three single edged controlled PWM outputs. (The number of match outputs for each timer that are actually available on device pins may vary by device.)
- Up to two match registers can be used to generate DMA requests. These are connected to DMA trigger inputs on this device.

25.3 Basic configuration

- Select clock source. See Section 4.5.29 “CTimer 0 clock source select” to Section 4.5.33 “CTimer 4 clock source select register”.

- Set the appropriate bits to enable clocks to timers that will be used AHBCLKCTRL registers, see Section 4.5.18 “AHB clock control 1” and Section 4.5.19 “AHB clock control 2”.
- Clear the timer reset using the PRESETCTRL registers, see Section 4.5.8 “Peripheral reset control 1” and Section 4.5.9 “Peripheral reset control 2”. Note that bit positions in the reset control registers match the bit positions in the clock control registers.
- Pins: Select timer pins and pin modes as needed through the relevant IOCON registers, see Chapter 15 “LPC55S6x/LPC55S2x/LPC552x I/O Pin Configuration (IOCON)”.
- Interrupts: See register MCR (Table 559) and CCR (Table 561) for match and capture events. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register. For interrupt connections, see Table 8.
- DMA: Some timer match conditions can be used to generate timed DMA requests, see Chapter 22 “LPC55S6x/LPC55S2x/LPC552x DMA controller”.

25.4 General description

Each Counter/timer is designed to count cycles of the APB bus clock 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 capture inputs 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. All match registers can optionally be auto-reloaded from a companion shadow register whenever the counter is reset to zero. This permits modifying the match values for the next counter cycle without risk of disrupting the PWM waveforms during the current cycle. When enabled, match reload will occur whenever the counter is reset either due to a match event or a write to bit 1 of the Timer Control Register (TCR).

25.4.1 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 generated by one of the pins with a capture function. 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 APB bus clock. For more details see Section 25.6.11 “Count control register”.

25.4.2 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 (PWMC) control the functionality of this output.

25.4.3 Applications

- Interval timer for counting internal events.

- Pulse Width Modulator via match outputs.
- Pulse Width Demodulator via capture input.
- Free running timer.

25.4.4 Architecture

The block diagram for the timers is shown in Figure 100.

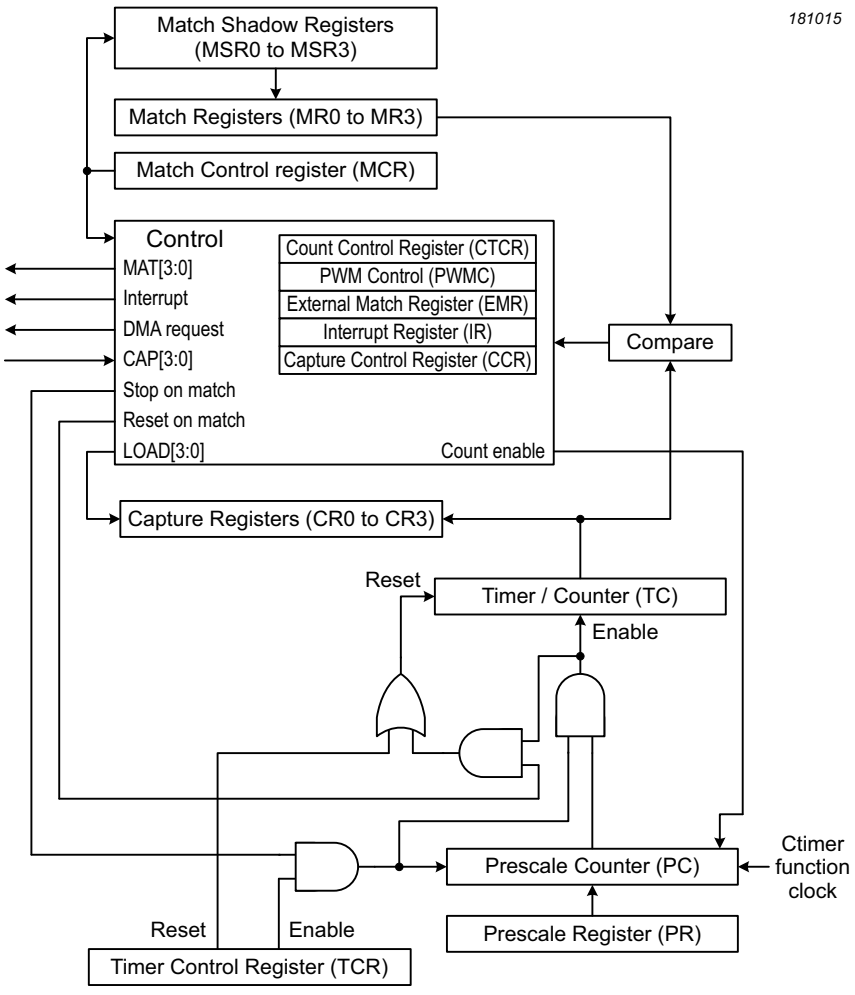


Fig 100. 32-bit counter/timer block diagram

25.4.5 Peripheral input multiplexers for CTimers

See Chapter 15 “LPC55S6x/LPC55S2x/LPC552x I/O Pin Configuration (IOCON)”.

The pen assignments for the peripheral input multiplexer is shown in figure Figure 101

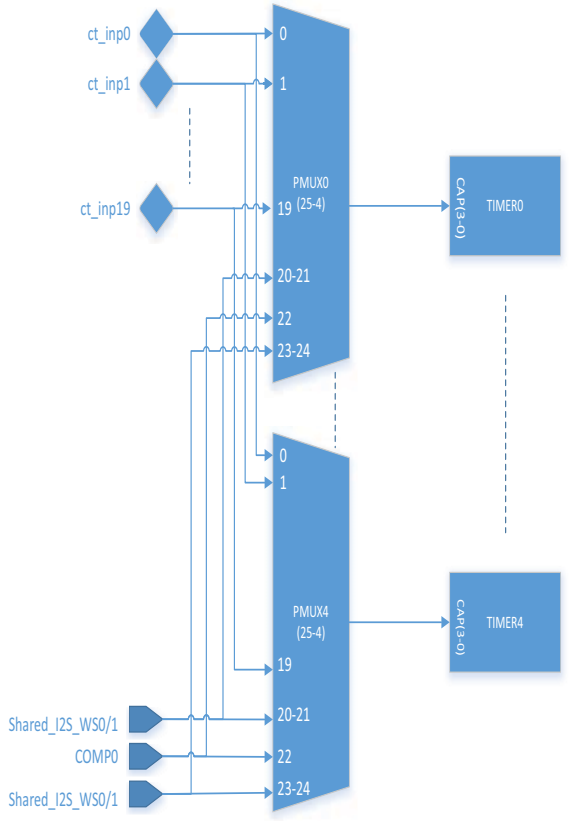


Fig 101. Pen assignments for the Peripheral input multiplexer for CTimers

25.5 Pin description

Table 552 gives a brief summary of each of the Timer/Counter related pins.

Table 552. Timer/Counter pin description

Pin	Type	Description
CTIMER0_CAP3:0 CTIMER1_CAP3:0 CTIMER2_CAP3:0 CTIMER3_CAP3:0 CTIMER4_CAP3:0	Input	Capture Signals- A transition on a capture pin can be configured to load one of the Capture registers with the value in the Timer Counter and optionally generate an interrupt. Capture functionality can be selected from a number of pins. Timer/Counter block can select a capture signal as a clock source instead of the APB bus clock. For more details see Section 25.6.11 “Count control register”.
CTIMER0_MAT3:0 CTIMER1_MAT3:0 CTIMER2_MAT3:0 CTIMER3_MAT3:0 CTIMER4_MAT0	Output	External Match Output - When a match register (MR3:0) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output. Match Output functionality can be selected on a number of pins in parallel.

25.5.1 Multiple CAP and MAT pins

Software can select from multiple pins for the CAP or MAT functions in the IOCON registers, which are described in Chapter 15 “LPC55S6x/LPC55S2x/LPC552x I/O Pin Configuration (IOCON)”. Note that match conditions may be used internally without the use of a device pin.

25.6 Register description

Each Timer/Counter contains the registers shown in Table 553.

Table 553. Register overview: CTIMER0/1/2/3 (register base addresses 0x4000 8000 (CTIMER0), 0x4000 9000 (CTIMER1), 0x4002 8000 (CTIMER2), 0x4002 9000 (CTIMER3), 0x4002 A000 (CTIMER4)

Name	Access	Offset	Description	Reset value[1]	Section
IR	R/W	0x00	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	25.6.1
TCR	R/W	0x04	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	25.6.2
TC	R/W	0x08	Timer Counter. The 32 bit TC is incremented every PR+1 cycles of the APB bus clock. The TC is controlled through the TCR.	0	25.6.3
PR	R/W	0x0C	Prescale Register. When the Prescale Counter (PC) is equal to this value, the next clock increments the TC and clears the PC.	0	25.6.4
PC	R/W	0x10	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	25.6.5
MCR	R/W	0x14	The MCR is used to control whether an interrupt is generated, whether the TC is reset when a Match occurs, and whether the match register is reloaded from its shadow register when the TC is reset.	0	25.6.6
MR0	R/W	0x18	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	25.6.7
MR1	R/W	0x1C	Match Register 1. See MR0 description.	0	25.6.7
MR2	R/W	0x20	Match Register 2. See MR0 description.	0	25.6.7
MR3	R/W	0x24	Match Register 3. See MR0 description.	0	25.6.7
CCR	R/W	0x28	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	25.6.8
CR0	RO	0x2C	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CAPn.0 input.	0	25.6.9
CR1	RO	0x30	Capture Register 1. See CR0 description.	0	25.6.9
CR2	RO	0x34	Capture Register 2. See CR0 description.	0	25.6.9
CR3	RO	0x38	Capture Register 3. See CR0 description.	0	25.6.9
EMR	R/W	0x3C	External Match Register. The EMR controls the match function and the external match pins.	0	25.6.10
CTCR	R/W	0x70	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0	25.6.11
PWMC	R/W	0x74	PWM Control Register. The PWMC enables PWM mode for the external match pins.	0	25.6.12
MSR0	R/W	0x78	Match 0 Shadow Register. If enabled, the Match 0 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero.	0	25.6.13

Table 553. Register overview: CTIMER0/1/2/3 (register base addresses 0x4000 8000 (CTIMER0), 0x4000 9000 (CTIMER1), 0x4002 8000 (CTIMER2), 0x4002 9000 (CTIMER3), 0x4002 A000 (CTIMER4) *…continued*

Name	Access	Offset	Description	Reset value[1]	Section
MSR1	R/W	0x7C	Match 1 Shadow Register. If enabled, the Match 1 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero.	0	25.6.13
MSR2	R/W	0x80	Match 2 Shadow Register. If enabled, the Match 2 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero.	0	25.6.13
MSR3	R/W	0x84	Match 3 Shadow Register. If enabled, the Match 3 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero.	0	25.6.13

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

25.6.1 Interrupt register

The Interrupt Register consists of 4 bits for the match interrupts and 4 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. The act of clearing an interrupt for a timer match also clears any corresponding DMA request. Writing a zero has no effect.

Table 554. Interrupt register (IR, offset 0x000)

Bit	Symbol	Description	Reset Value
0	MR0INT	Interrupt flag for match channel 0.	0
1	MR1INT	Interrupt flag for match channel 1.	0
2	MR2INT	Interrupt flag for match channel 2.	0
3	MR3INT	Interrupt flag for match channel 3.	0
4	CR0INT	Interrupt flag for capture channel 0 event.	0
5	CR1INT	Interrupt flag for capture channel 1 event.	0
6	CR2INT	Interrupt flag for capture channel 2 event.	0
7	CR3INT	Interrupt flag for capture channel 3 event.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.2 Timer control register

The Timer Control Register (TCR) is used to control the operation of the Timer/Counter.

Table 555. Timer Control Register (TCR, offset 0x004)

Bit	Symbol	Value	Description	Reset value
0	CEN		Counter enable.	0
		0	Disabled.The counters are disabled.	
		1	Enabled. The timer counter and pre-scale counter are enabled.	

Table 555. Timer Control Register (TCR, offset 0x004) *…continued*

Bit	Symbol	Value	Description	Reset value
1	CRST		Counter reset.	0
		0	Disabled. Do nothing.	
		1	Enabled. The Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of the APB bus clock. The counters remain reset until TCR[1] is returned to zero.	
31:2	-	-	Reserved. Read value is undefined, only zero should be written.	NA

25.6.3 Timer counter registers

The 32-bit timer counter register is incremented when the prescale counter reaches its terminal count. Unless it is reset before reaching its upper limit, the Timer Counter 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 556. Timer counter registers (TC, offset 0x08)

Bit	Symbol	Description	Reset value
31:0	TCVAL	Timer counter value.	0

25.6.4 Pre-scale register

The 32-bit Pre-scale register specifies the maximum value for the Pre-scale Counter.

Table 557. Timer pre scale registers (PR, offset 0x00C)

Bit	Symbol	Description	Reset value
31:0	PRVAL	Pre-scale counter value.	0

25.6.5 Pre-scale counter register

The 32-bit pre-scale counter controls division of the APB bus clock by some constant value before it is applied to the timer counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The pre-scale counter is incremented on every APB bus clock. When it reaches the value stored in the pre-scale register, the timer counter is incremented and the pre-scale counter is reset on the next APB bus clock. This causes the timer counter to increment on every APB bus clock when PR = 0, every 2 APB bus clocks when PR = 1, etc.

Table 558. Timer pre-scale counter registers (PC, offset 0x010)

Bit	Symbol	Description	Reset value
31:0	PCVAL	Pre-scale counter value.	0

25.6.6 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.

Table 559. Match Control Register (MCR, offset 0x014)

Bit	Symbol	Description	Reset Value
0	MR0I	Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC. 0 = disabled. 1 = enabled.	0
1	MR0R	Reset on MR0: the TC will be reset if MR0 matches it. 0 = disabled. 1 = enabled.	0
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 = disabled. 1 = enabled.	0
3	MR1I	Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC. 0 = disabled. 1 = enabled. 0 = disabled. 1 = enabled.	0
4	MR1R	Reset on MR1: the TC will be reset if MR1 matches it. 0 = disabled. 1 = enabled.	0
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 = disabled. 1 = enabled.	0
6	MR2I	Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC. 0 = disabled. 1 = enabled.	0
7	MR2R	Reset on MR2: the TC will be reset if MR2 matches it. 0 = disabled. 1 = enabled.	0
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 = disabled. 1 = enabled.	0
9	MR3I	Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC. 0 = disabled. 1 = enabled.	0
10	MR3R	Reset on MR3: the TC will be reset if MR3 matches it. 0 = disabled. 1 = enabled.	0
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 = disabled. 1 = enabled.	0
23:12	-	Reserved. Read value is undefined, only zero should be written.	NA
24	MR0RL	Reload MR0 with the contents of the Match 0 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled.	0
25	MR1RL	Reload MR1 with the contents of the Match 1 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled.	0
26	MR2RL	Reload MR2 with the contents of the Match 2 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled.	0
27	MR3RL	Reload MR3 with the contents of the Match 3 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled.	0
31:28	-	Reserved. Read value is undefined, only zero should be written.	NA

25.6.7 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.

If the associated MRxRL bit in the match control register is set, the match register will be automatically reloaded with the current contents of its corresponding match shadow register whenever the TC is cleared to zero. This transfer will take place on the same clock edge that clocks the TC to zero.

Note: The TC is typically reset in response to an occurrence of a match on the Match Register being used to set the cycle counter rate. A reset can also occur due to software writing a 1 to bit 1 of the timer control register.

Table 560. Timer match registers (MR[0:3], offset [0x018:0x024])

Bit	Symbol	Description	Reset value
31:0	MATCH	Timer counter match value.	0

25.6.8 Capture control register

The Capture control register is used to control whether one of the four capture registers is loaded with the value in the timer counter when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. In the description below, *n* represents the timer number, 0 or 1.

Note: If counter mode is selected for a particular CAP input in the CTCR, the three bits for that input in this register should be programmed as 000, but capture and/or interrupt can be selected for the other three CAP inputs.

Table 561. Capture control register (CCR, offset 0x028)

Bit	Symbol	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 = disabled. 1 = enabled.	0
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 = disabled. 1 = enabled.	0
2	CAP0I	Generate interrupt on channel 0 capture event: a CR0 load generates an interrupt.	0
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 = disabled. 1 = enabled.	0
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 = disabled. 1 = enabled.	0
5	CAP1I	Generate interrupt on channel 1 capture event: a CR1 load generates an interrupt.	0
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 = disabled. 1 = enabled.	0
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 = disabled. 1 = enabled.	0
8	CAP2I	Generate interrupt on channel 2 capture event: a CR2 load generates an interrupt.	0
9	CAP3RE	Rising edge of capture channel 3: a sequence of 0 then 1 causes CR3 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
10	CAP3FE	Falling edge of capture channel 3: a sequence of 1 then 0 causes CR3 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
11	CAP3I	Generate interrupt on channel 3 capture event: a CR3 load generates an interrupt.	0
31:12	-	Reserved. Read value is undefined, only zero should be written.	NA

25.6.9 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 could originate from an external pin or from an internal source. The

settings in the capture control 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 562. Timer capture registers (CR[0:3], offsets [0x02C:0x038])

Bit	Symbol	Description	Reset value
31:0	CAP	Timer counter capture value.	0

25.6.10 External match register

The External match register provides both control and status of the external match pins. In the following descriptions, *n* represents the timer number, 0 or 1, and *m* represents a match number, 0 through 3.

Match events for Match 0 and Match 1 in each timer can cause a DMA request, see Section 25.7.2 “DMA operation (DMA0 and DMA1)”.

If the match outputs are configured as PWM output, the function of the external match registers is determined by the PWM rules, see Section 25.7.1 “Rules for single edge controlled PWM outputs”.

Table 563. Timer external match registers (EMR, offset 0x03C)

Bit	Symbol	Value	Description	Reset value
0	EM0	-	External Match 0. This bit reflects the state of output MAT0, whether or not this output is connected to a pin. When a match occurs between the TC and MR0, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[5:4]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0
1	EM1	-	External Match 1. This bit reflects the state of output MAT1, whether or not this output is connected to a pin. When a match occurs between the TC and MR1, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[7:6]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0
2	EM2	-	External Match 2. This bit reflects the state of output MAT2, whether or not this output is connected to a pin. When a match occurs between the TC and MR2, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[9:8]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0
3	EM3	-	External Match 3. This bit reflects the state of output MAT3, whether or not this output is connected to a pin. When a match occurs between the TC and MR3, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by MR[11:10]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0
5:4	EMC0		External Match Control 0. Determines the functionality of External Match 0.	00
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (MAT0 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (MAT0 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	

Table 563. Timer external match registers (EMR, offset 0x03C) *…continued*

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 (MAT1 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (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 (MAT2 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (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 (MAT3 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (MAT3 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
31:12	-	-	Reserved. Read value is undefined, only zero should be written.	NA

25.6.11 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 edge(s) for counting.

When counter mode is chosen as a mode of operation, the CAP input (selected by the CTCR bits 3:2) is sampled on every rising edge of the APB bus 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. The timer counter register is incremented only if the identified event occurs and the event corresponds to the one selected by bits 1:0 in the CTCR register.

Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the APB bus 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 APB bus clock. Consequently, duration of the HIGH/LOWLOW levels on the same CAP input in this case cannot be shorter than 1/APB bus clock.

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 564. Count Control Register (CTCR, offset 0x070)

Bit	Symbol	Value	Description	Reset Value
1:0	CTMODE		Counter/Timer mode This field selects which rising APB bus clock edges can increment timer's pre-scale Counter (PC), or clear PC and increment Timer Counter (TC). Timer mode: the TC is incremented when the pre-scale counter matches the pre-scale register.	00
		0x0	Timer mode. Incremented every rising APB bus clock 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	CINSEL		Count input select When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking. **Note:** If counter mode is selected for a particular CAPn input in the CTCR, the three bits for that input in the Capture Control Register (CCR) must be programmed as 000. However, capture and/or interrupt can be selected for the other three CAPn inputs in the same timer.	0
		0x0	Channel 0. CAPn.0 for CTIMERn	
		0x1	Channel 1. CAPn.1 for CTIMERn	
		0x2	Channel 2. CAPn.2 for CTIMERn	
		0x3	Channel 3. CAPn.3 for CTIMERn	
4	ENCC	-	Setting this bit to 1 enables clearing of the timer and the pre-scaler when the capture-edge event specified in bits 7:5 occurs.	0
7:5	SELCC		Edge select. When bit 4 is 1, these bits select which capture input edge will cause the timer and pre-scaler to be cleared. These bits have no effect when bit 4 is low. Note that different part number and package variations may provide different capture input pin functions.	0
		0x0	Channel 0 rising edge. Rising edge of the signal on capture channel 0 clears the timer (if bit 4 is set).	
		0x1	Channel 0 falling edge. Falling edge of the signal on capture channel 0 clears the timer (if bit 4 is set).	
		0x2	Channel 1 rising edge. Rising edge of the signal on capture channel 1 clears the timer (if bit 4 is set).	
		0x3	Channel 1 falling edge. Falling edge of the signal on capture channel 1 clears the timer (if bit 4 is set).	
		0x4	Channel 2 rising edge. Rising edge of the signal on capture channel 2 clears the timer (if bit 4 is set).	
		0x5	Channel 2 falling edge. Falling edge of the signal on capture channel 2 clears the timer (if bit 4 is set).	
		0x6	Channel 3 rising edge. Rising edge of the signal on capture channel 3 clears the timer (if bit 4 is set).	
		0x7	Channel 3 falling edge. Falling edge of the signal on capture channel 3 clears the timer (if bit 4 is set).	
31:8	-	-	Reserved. Read value is undefined, only zero should be written.	NA

25.6.12 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 565. PWM control register (PWMC, offset 0x074)

Bit	Symbol	Value	Description	Reset value
0	PWMEN0		PWM mode enable for channel0.	0
		0	Match. CTIMERn_MAT0 is controlled by EM0.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT0.	
1	PWMEN1		PWM mode enable for channel1.	0
		0	Match. CTIMERn_MAT01 is controlled by EM1.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT1.	
2	PWMEN2		PWM mode enable for channel2.	0
		0	Match. CTIMERn_MAT2 is controlled by EM2.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT2.	
3	PWMEN3		PWM mode enable for channel3. **Note:** It is recommended to use match channel 3 to set the PWM cycle.	0
		0	Match. CTIMERn_MAT3 is controlled by EM3.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT3.	
31:4	-		Reserved. Read value is undefined, only zero should be written.	NA

25.6.13 Match shadow registers

The Match shadow registers contain the values that the corresponding Match registers are (optionally) reloaded with at the start of each new counter cycle. Typically, the match that causes the counter to be reset (and instigates the match reload) will also be programmed to generate an interrupt or DMA request. Software or the DMA engine will then have one full counter cycle to modify the contents of the Match Shadow Register(s) before the next reload occurs.

Table 566. Timer match shadow registers (MSR[0:3], offset [0x78:0x84])

Bit	Symbol	Description	Reset value
31:0	SHADOW	Timer counter match shadow value.	0x0

25.7 Functional description

Figure 102 shows a timer configured to reset the count and generate an interrupt on match. The pre-scaler is set to two and the match register set to six. 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 103 shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to two and the match register set to six. 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.

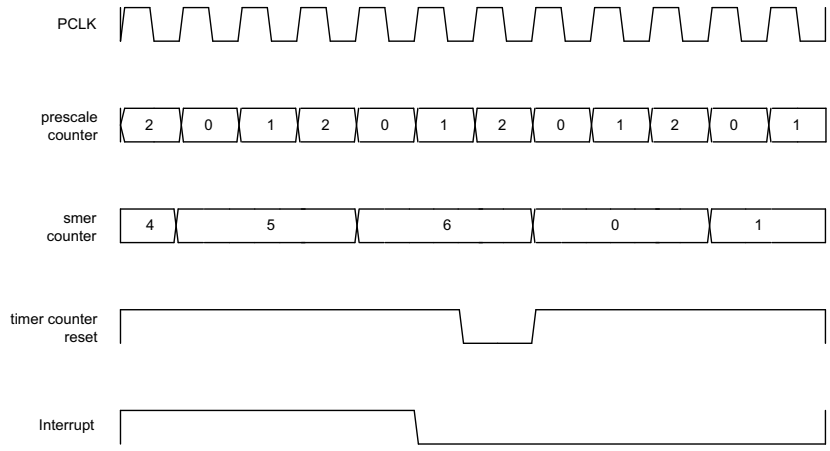


Fig 102. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

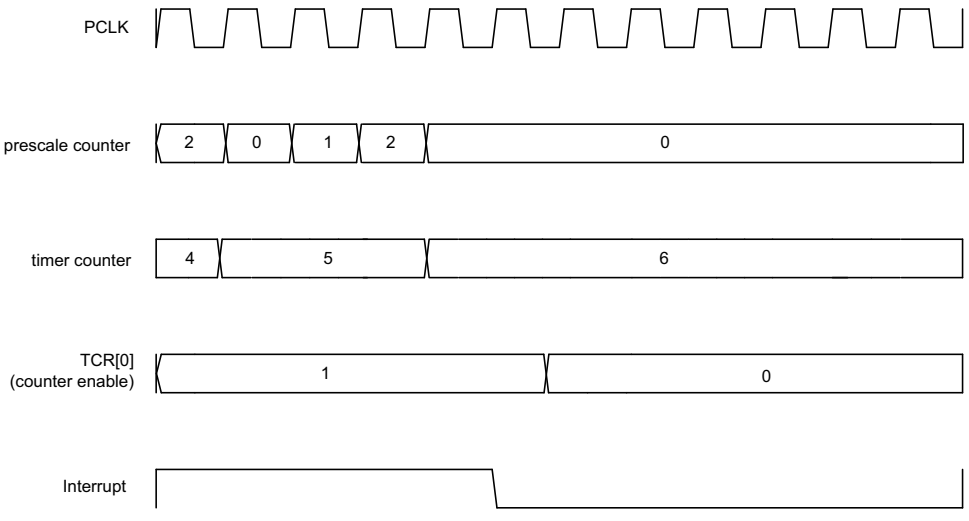


Fig 103. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

25.7.1 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 (that is, 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 (that is, 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.

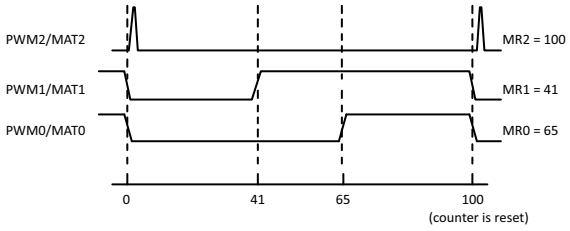


Fig 104. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

25.7.2 DMA operation (DMA0 and DMA1)

DMA requests are generated by a match of the Timer Counter (TC) register value to either Match Register 0 (MR0) or Match Register 1 (MR1). This is not connected to the operation of the Match outputs controlled by the EMR register. Each match sets a DMA trigger flag, which is connected to the DMA controller. In order to have an effect, the DMA controller must be configured correctly.

When a timer is initially set up to generate a DMA request, the request may already be asserted before a match condition occurs. An initial DMA request may be avoided by having software write a one to the interrupt flag location, as if clearing a timer interrupt. See Section 25.6.1 “Interrupt register”. A DMA request is cleared automatically when it is a handled by the DMA controller.

Note: Because timer DMA requests are generated whenever the timer value is equal to the related Match Register value, DMA requests are always generated when the timer is running, unless the Match Register value is higher than the upper count limit of the timer. It is important not to select and enable timer DMA requests in the DMA block unless the timer is correctly configured to generate valid DMA requests.

26.1 How to read this chapter

The Micro-tick timer is available on all LPC55S6x/LPC55S2x/LPC552x devices.

26.2 Features

- Ultra simple, ultra-low power timer that can run and wake up the device in reduced power modes other than power-down and deep-power down.
- Write once to start.
- Interrupt or software polling.
- Four capture registers that can be triggered by external pin transitions.

26.3 Basic configuration

Configure the Micro-tick timer as follows:

- Set the UTICK bit in the AHBCLKCTRL1 register to enable the clock to the Micro-tick Timer register interface.
- The Micro-tick Timer provides an interrupt to the NVIC, see Chapter 3 “LPC55S6x/LPC55S2x/LPC552x Nested Vectored Interrupt Controller (NVIC)”.
- To enable Micro-tick timer interrupts for waking up from deep-sleep, use the low power API provided Power_EnterDeepSleep. See Chapter 14 “LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API” on page 312.
- Configure the pin functions of any Micro-tick timer capture pins that will be used via IOCON, see Chapter 15 “LPC55S6x/LPC55S2x/LPC552x I/O Pin Configuration (IOCON)”.
- Enable 1MHz clock by programming bit 4 in Syscon CLOCK_CTRL register. See Section 4.5.75 “Clock control”.

26.4 General description

Figure 105 shows a conceptual view of the Micro-tick timer.

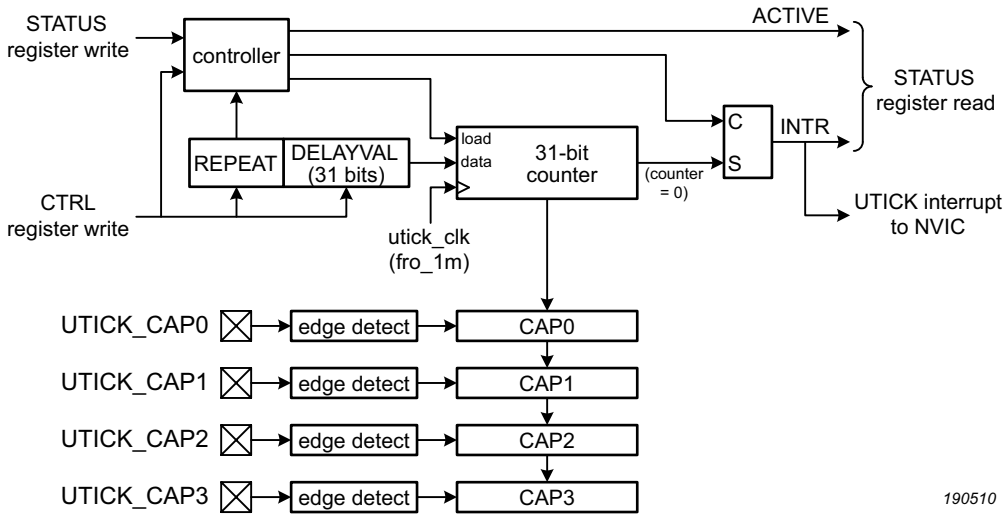


Fig 105. MIcro-tick timer block diagram

26.5 Pin description

Table 567 gives a summary of pins related to the Micro-tick Timer.

Table 567. Micro-tick Timer pin description

Pin	Port Pins	Type	Description
UTICK_CAP0, UTICK_CAP1, UTICK_CAP2, UTICK_CAP3	PIO0_13, PIO0_14, PIO0_15, PIO0_19, PIO0_21, PIO0_22, PIO1_14, PIO1_15, PIO1_25, PIO1_26	Input	Capture inputs. The selected transition on a capture pin can be configured to load the related CAP register with the value of counter.

26.6 Register description

The Micro-tick Timer contains the registers shown in Table 568. Note that the Micro-tick Timer operates from a different (typically slower) clock than the CPU and bus systems. This means there may be a synchronization delay when accessing Micro-tick Timer registers.

Table 568. Register overview: Micro-tick Timer (base address = 0x5000 E000)

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	Control register.	0	26.6.1
STAT	R/W	0x004	Status register.	0	26.6.2
CFG	R/W	0x008	Capture configuration register.	0	26.6.3
CAPCLR	WO	0x00C	Capture clear register.	NA	26.6.4
CAP0	RO	0x010	Capture register 0.	0	26.6.5
CAP1	RO	0x014	Capture register 1.	0	26.6.5
CAP2	RO	0x018	Capture register 2.	0	26.6.5
CAP3	RO	0x01C	Capture register 3.	0	26.6.5

26.6.1 CTRL register

This register controls the Micro-tick timer. Any write to the CTRL register resets the counter, meaning a new interval will be measured if one was in progress.

Table 569. Control register (CTRL, offset = 0x000)

Bit	Symbol	Description	Reset value
30:0	DELAYVAL	Tick interval value. The delay will be equal to DELAYVAL + 1 periods of the timer clock. The minimum usable value is 1, for a delay of 2 timer clocks. A value of 0 stops the timer.	0
31	REPEAT	Repeat delay. 0 = One-time delay. 1 = Delay repeats continuously.	0

26.6.2 Status register

This register provides status for the Micro-tick Timer.

Table 570. Status register (STAT, offset = 0x004)

Bit	Symbol	Description	Reset value
0	INTR	Interrupt flag. 0 = No interrupt is pending. 1 = An interrupt is pending. A write of any value to this register clears this flag.	0
1	ACTIVE	Active flag. 0 = The Micro-tick Timer is stopped. 1 = The Micro-tick Timer is currently active.	0
31:2	-	Reserved	-

26.6.3 Capture configuration register

This register allows enabling Micro-tick capture functions and selects the polarity of the capture triggers.

Table 571. Capture configuration register (CFG, offset = 0x008)

Bit	Symbol	Description	Reset value
0	CAPEN0	Enable capture 0. 1 = Enabled, 0 = Disabled.	0
1	CAPEN1	Enable capture 1. 1 = Enabled, 0 = Disabled.	0
2	CAPEN2	Enable capture 2. 1 = Enabled, 0 = Disabled.	0
3	CAPEN3	Enable capture 3. 1 = Enabled, 0 = Disabled.	0
7:4	-	Reserved	-
8	CAPPOL0	Capture polarity 0. 0 = Positive edge capture, 1 = Negative edge capture.	0
9	CAPPOL1	Capture polarity 1. 0 = Positive edge capture, 1 = Negative edge capture.	0
10	CAPPOL2	Capture polarity 2. 0 = Positive edge capture, 1 = Negative edge capture.	0
11	CAPPOL3	Capture polarity 3. 0 = Positive edge capture, 1 = Negative edge capture.	0
31:12	-	Reserved	-

26.6.4 Capture clear register

This read-only register allows clearing previous capture values, allowing new captures to take place.

Table 572. Capture clear register (CAPCLR, offset = 0x00C)

Bit	Symbol	Description	Reset value
0	CAPCLR0	Clear capture 0. Writing 1 to this bit clears the CAP0 register value.	NA
1	CAPCLR1	Clear capture 1. Writing 1 to this bit clears the CAP1 register value.	NA
2	CAPCLR2	Clear capture 2. Writing 1 to this bit clears the CAP2 register value.	NA
3	CAPCLR3	Clear capture 3. Writing 1 to this bit clears the CAP3 register value.	NA
31:4	-	Reserved	-

26.6.5 Capture registers

This register contains the Micro-tick timer value based on any previously capture events. Each capture register is associated with one of the capture trigger inputs.

Table 573. Capture registers (CAP[0:3], offsets = [0x010:0x01C])

Bit	Symbol	Description	Reset value
30:0	CAP_VALUE	Capture value for the related capture event (UTICK_CAPn. Note: The value 1 is lower than the actual value of the Micro-tick Timer at the moment of the capture event.	0
31	VALID	Capture valid. When 1, a value has been captured based on a transition of the related UTICK_CAPn pin. Cleared by writing to the related bit in the CAPCLR register.	0

27.1 How to read this chapter

The MRT is available on all LPC55S6x/LPC55S2x/LPC552x devices.

27.2 Features

- 24-bit interrupt timer.
- Four channels independently counting down from individually set values.
- Repeat interrupt, one-shot interrupt, and one-shot bus stall modes.

27.3 Basic configuration

Configuration of the MRT is accomplished as followings

- In the AHBCLKCTRL1 register, see Table 124, set the MRT bit to enable the clock to the register interface.
- Clear the MRT reset using the PRESETCTRL1 register, see Table 114.
- The global MRT interrupt is connected to an interrupt slot in the NVIC, see Table 72.
- It is recommended that the MRT counters are stopped before entering in deep-sleep low power mode (before calling the low power API Power_EnterDeepSleep()). The Power_EnterDeepSleep() API modifies the System Clock frequency, which impacts any MRT counter that is running.

27.4 Pin description

The MRT is not associated with any device pins.

27.5 General description

The Multi-Rate Timer (MRT) provides a repetitive interrupt timer with four channels. Each channel can be programmed with an independent time interval.

- Repeat interrupt mode. See Section 27.5.1 “Repeat interrupt mode”
- One-shot interrupt mode. See Section 27.5.2 “One-shot interrupt mode”
- One-shot stall mode. See Section 27.5.3 “One-shot stall mode”

The modes for each timer are set in the timer’s control register. See Table 577.

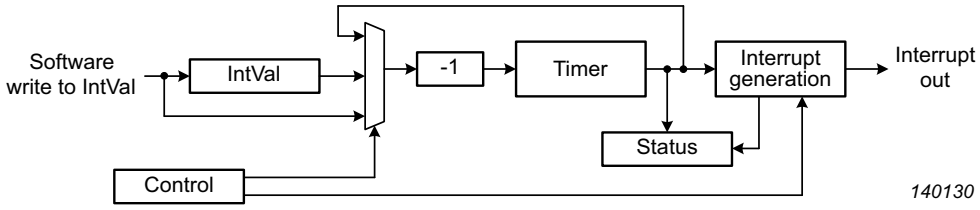


Fig 106. MRT block diagram

27.5.1 Repeat interrupt mode

The repeat interrupt mode generates repeated interrupts after a selected time interval. This mode can be used for software-based PWM or PPM applications.

When the timer n is in idle state, writing a non-zero value IVALUE to the INTVALn register immediately loads the time interval value IVALUE - 1, and the timer begins to count down from this value. When the timer reaches zero, an interrupt is generated, the value in the INTVALn register IVALUE - 1 is reloaded automatically, and the timer starts to count down again.

While the timer is running in repeat interrupt mode, the following actions can be performed

- Change the interval value on the next timer cycle by writing a new value (>0) to the INTVALn register and setting the LOAD bit to 0. An interrupt is generated when the timer reaches zero. On the next cycle, the timer counts down from the new value.
- Change the interval value real-time immediately by writing a new value (>0) to the INTVALn register and setting the LOAD bit to 1. The timer immediately starts to count down from the new timer interval value. An interrupt is generated when the timer reaches 0.
- Stop the timer at the end of time interval by writing a 0 to the INTVALn register and setting the LOAD bit to 0. An interrupt is generated when the timer reaches zero.
- Stop the timer immediately by writing a 0 to the INTVALn register and setting the LOAD bit to 1. No interrupt is generated when the INTVALn register is written.

27.5.2 One-shot interrupt mode

The one-shot interrupt generates one interrupt after a one-time count. With this mode, a single interrupt can be generated at any point. This mode can be used to introduce a specific delay in a software task.

When the timer is in the idle state, writing a non-zero value IVALUE to the INTVALn register immediately loads the time interval value IVALUE - 1, and the timer starts to count down. When the timer reaches 0, an interrupt is generated and the timer stops and enters the idle state.

While the timer is running in the one-shot interrupt mode, the following actions can be performed:

- Update the INTVALn register with a new time interval value (>0) and set the LOAD bit to 1. The timer immediately reloads the new time interval, and starts counting down from the new value. No interrupt is generated when the TIME_INTVALn register is updated.
- Write a 0 to the INTVALn register and set the LOAD bit to 1. The timer immediately stops counting and moves to the idle state. No interrupt is generated when the INTVALn register is updated.

27.5.3 One-shot stall mode

One-shot stall mode is similar to one-shot interrupt mode, except that it is intended for very short delays, for instance when the delay needed is less than the time it takes to get to an interrupt service routine. This mode is designed for very low software overhead,

requiring only a single write to the INTVAL register (if the channel is already configured for one-shot stall mode). The MRT times the requested delay while stalling the bus write operation, concluding the write when the delay is complete. No interrupt or status polling is needed.

Bus stall mode can be used when a short delay is need between two software controlled events, or when a delay is expected before software can continue. Since in this mode there are no bus transactions while the MRT is counting down, the CPU consumes a minimum amount of power during that time.

Note that bus stall mode provides a minimum amount of time between the execution of the instruction that performs the write to INTVAL and the time that software continues. Other system events, such as interrupts or other bus masters accessing the APB bus where the MRT resides, can cause the delay to be longer.

27.6 Register description

The reset values in Table 574 are POR reset values.

Table 574. Register overview: MRT (base address = 0x4000 D000)

Name	Access	Offset	Description	Reset value	Section
MRT Timer 0 registers					
INTVAL0	R/W	0x0	MRT0 time interval value. This value is loaded into the TIMER0 register.	0	27.6.1
TIMER0	RO	0x4	MRT0 timer. This register reads the value of the down-counter.	0xFF FFFF	27.6.2
CTRL0	R/W	0x8	MRT0 control. This register controls the MRT0 modes.	0	27.6.3
STAT0	R/W	0xC	MRT0 status.	0	27.6.4
MRT Timer 1 registers					
INTVAL1	R/W	0x10	MRT1 time interval value. This value is loaded into the TIMER1 register.	0	27.6.1
TIMER1	RO	0x14	MRT1 timer. This register reads the value of the down-counter.	0xFF FFFF	27.6.2
CTRL1	R/W	0x18	MRT1 control. This register controls the MRT0 modes.	0	27.6.3
STAT1	R/W	0x1C	MRT1 Status.	0	27.6.4
MRT Timer 2 registers					
INTVAL2	R/W	0x20	MRT2 time interval value. This value is loaded into the TIMER2 register.	0	27.6.1
TIMER2	RO	0x24	MRT2 timer. This register reads the value of the down-counter.	0xFF FFFF	27.6.2
CTRL2	R/W	0x28	MRT2 control. This register controls the MRT0 modes.	0	27.6.3
STAT2	R/W	0x2C	MRT2 status.	0	27.6.4
MRT Timer 3 registers					
INTVAL3	R/W	0x30	MRT3 time interval value. This value is loaded into the TIMER3 register.	0	27.6.1
TIMER3	RO	0x34	MRT3 timer. This register reads the value of the down-counter.	0xFF FFFF	27.6.2
CTRL3	R/W	0x38	MRT3 control. This register controls the MRT0 modes.	0	27.6.3
STAT3	R/W	0x3C	MRT3 status.	0	27.6.4
Global MRT registers					
MODCFG	R/W	0xF0	Module configuration. This register provides information about this particular MRT instance, and allows choosing an overall mode for the idle channel feature.	0	27.6.5
IDLE_CH	RO	0xF4	Idle channel. This register returns the number of the first idle channel.	0	27.6.6
IRQ_FLAG	R/W	0xF8	Global interrupt flag.	0	27.6.7

27.6.1 Time interval register

This register contains the MRT load value and controls how the timer is reloaded. The load value is IVALUE -1.

Table 575. Time interval register (INTVAL[0:3], offset = 0x000 (INTVAL0) to 0x030 (INTVAL3))

Bit	Symbol	Value	Description	Reset value
23:0	IVALUE		Time interval load value. This value is loaded into the TIMERn register and the MRT channel n starts counting down from IVALUE -1. If the timer is idle, writing a non-zero value to this bit field starts the timer immediately. If the timer is running, writing a zero to this bit field does the following: • If LOAD = 1, the timer stops immediately. • If LOAD = 0, the timer stops at the end of the time interval.	0
30:24	-	-	Reserved. Read value is undefined, only zero should be written.	-
31	LOAD		Determines how the timer interval value IVALUE -1 is loaded into the TIMERn register. This bit is write-only. Reading this bit always returns 0.	0
		0	No force load. The load from the INTVALn register to the TIMERn register is processed at the end of the time interval if the repeat mode is selected.	
		1	Force load. The INTVALn interval value IVALUE -1 is immediately loaded into the TIMERn register while TIMERn is in active state.	

27.6.2 Timer register

The timer register holds the current timer value. This register is read-only.

Table 576. Timer register (TIMER[0:3], offset = 0x004 (TIMER0) to 0x034 (TIMER3))

Bit	Symbol	Value	Description	Reset value
23:0	VALUE		Holds the current timer value of the down-counter. The initial value of the TIMERn register is loaded as IVALUE - 1 from the INTVALn register either at the end of the time interval or immediately in the following cases: INTVALn register is updated in the idle state. INTVALn register is updated with LOAD = 1. When the timer is in idle state, reading this bit fields returns -1 (0x00FF FFFF).	
31:24	-	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.3 Control register

The control register configures the mode for each MRT and enables the interrupt.

Table 577. Control register (CTRL[0:3], offset = 0x08 (CTRL0) to 0x38 (CTRL3))

Bit	Symbol	Value	Description	Reset value
0	INTEN		Enable the TIMERn interrupt.	0
		0	Disabled. TIMERn interrupt is disabled.	
		1	Enabled. TIMERn interrupt is enabled.	

Table 577. Control register (CTRL[0:3], offset = 0x08 (CTRL0) to 0x38 (CTRL3)) *...continued*

Bit	Symbol	Value	Description	Reset value
2:1	MODE		Selects timer mode.	0
		0x0	Repeat interrupt mode.	
		0x1	One-shot interrupt mode.	
		0x2	One-shot stall mode.	
		0x3	Reserved.	
31:3	-	-	Reserved.	-

27.6.4 Status register

This register indicates the status of each MRT.

Table 578. Status register (STAT[0:3], offset = 0x0C (STAT0) to 0x3C (STAT3))

Bit	Symbol	Value	Description	Reset value
0	INTFLAG		Monitors the interrupt flag.	0
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMERn has reached the end of the time interval. If the INTEN bit in the CONTROLn is also set to 1, the interrupt for timer channel n and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
1	RUN		Indicates the state of TIMERn. This bit is read-only.	0
		0	Idle state. TIMERn is stopped.	
		1	Active state. TIMERn is running.	
2	INUSE		Channel In Use flag. Operating details depend on the MULTITASK bit in the MODCFG register, and affects the use of IDLE_CH.	0
		0	This channel **is not** in use. A write '0' to this bit is no operation.	
		1	This channel **is** in use. A write '1' to this bit clears BOOK_CH to free up the channel resource booking.	
31:3	-	-	Reserved.	-

27.6.5 Module configuration register

The MODCFG register provides the configuration (number of channels and timer width) for this MRT.

Table 579. Module configuration register (MODCFG, offset = 0xF0)

Bit	Symbol	Value	Description	Reset value
3:0	NOC	-	Identifies the number of channels in this MRT. (4 channels on this device.)	0x3
8:4	NOB	-	Identifies the number of timer bits in this MRT. (24 bits wide on this device.)	0x17
30:9	-	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 579. Module configuration register (MODCFG, offset = 0xF0) *…continued*

Bit	Symbol	Value	Description	Reset value
31	MULTITASK		Selects the operating mode for the INUSE flags and the IDLE_CH register.	0
		0	Hardware status mode. In this mode, the INUSE(n) flags for all channels are reset.	
		1	Multi-task mode.	

27.6.6 Idle channel register

The idle channel register can be used to assist software in finding available channels in the MRT. This allows more flexibility by not giving hard assignments to software that makes use of the MRT, without the need to search for an available channel. Generally, IDLE_CH returns the lowest available channel number.

IDLE_CH can be used in two ways, controlled by the value of the MULTITASK bit in the MODCFG register. MULTITASK affects both the function of IDLE_CH, and the function of the INUSE bit for each MRT channel as follows:

- MULTITASK = 0: hardware status mode. The INUSE flags for all MRT channels are reset. IDLECH returns the lowest idle channel number. A channel is considered idle if its RUN flag = 0, and there is no interrupt pending for that channel.
- MULTITASK = 1: multi-task mode. In this mode, the INUSE flags allow more control over when MRT channels are released for further use. When IDLE_CH is read, returning a channel number of an idle channel, the INUSE flag for that channel is set by hardware. That channel will not be considered idle until its RUN flag = 0, there is no interrupt pending, and its INUSE flag = 0. This allows reserving an MRT channel with a single register read, and no need to start the channel before it is no longer considered idle by IDLE_CH. It also allows software to identify a specific MRT channel that it can use, then use it more than once without releasing it, removing the need to ask for an available channel for every use.

Table 580. Idle channel register (IDLE_CH, offset 0xF4)

Bit	Symbol	Value	Description	Reset value
3:0	-	-	Reserved.	-
7:4	CHAN	-	Idle channel. Reading the CHAN bits, returns the lowest idle timer channel. The number is positioned such that it can be used as an offset from the MRT base address in order to access the registers for the allocated channel. If all timer channels are running, CHAN = 0xF. See text above for more details	0
31:8	-	-	Reserved.	-

27.6.7 Global interrupt flag register

The global interrupt register combines the interrupt flags from the individual timer channels in one register. Setting and clearing each flag behaves in the same way as setting and clearing the INTFLAG bit in each of the STATUSn registers.

Table 581. Global interrupt flag register (IRQ_FLAG, offset 0xF8)

Bit	Symbol	Value	Description	Reset value
0	GFLAG0	-	Monitors the interrupt flag of TIMER0.	0x3
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMER0 has reached the end of the time interval. If the INTEN bit in the CONTROL0 register is also set to 1, the interrupt for timer channel 0 and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
1	GFLAG1	-	Monitors the interrupt flag of TIMER1. See description of channel 0.	0
2	GFLAG2	-	Monitors the interrupt flag of TIMER1. See description of channel 0.	0
3	GFLAG3	-	Monitors the interrupt flag of TIMER1. See description of channel 0.	0
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	0

28.1 How to read this chapter

The RTC is available on all LPC55S6x/LPC55S2x/LPC552x devices.

28.2 Features

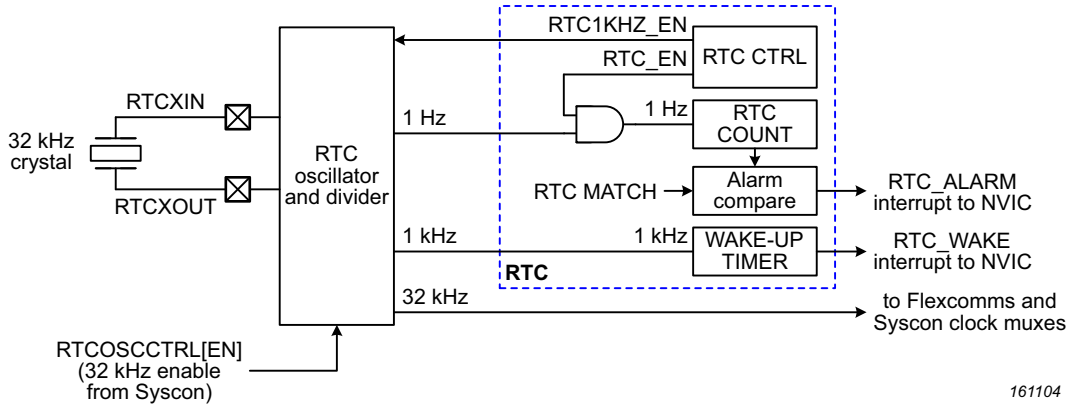
- The RTC oscillator has the following clock outputs:
  - 32.768 kHz clock (named as 32 kHz clock in rest of this chapter) 32 kHz clock, selectable for system clock and CLKOUT pin.The 32-kHz clock can be either the 32-kHz XTAL or the 32-kHz Free Running Oscillator.
  - 1 Hz clock for RTC timing.
  - 1024 Hz clock (named as 1 kHz clock in rest of this chapter) for high-resolution RTC timing.
- 32-bit, 1 Hz RTC counter and associated match register for alarm generation.
- 15-bit, 32 kHz sub-second counter.
- Separate 16-bit high-resolution/wake-up timer clocked at 1 kHz for 1 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 that go to one NVIC channel. Either time-out can wake up the part from any of the low power modes, including deep power-down.
- Eight 32-bit general purpose registers can retain data in deep power-down or in the event of a power failure, provided there is battery backup.

28.3 Basic configuration

Configure the RTC as follows:

- Use the AHBCLKCTRL0 register, see Section 4.5.17 “AHB clock control 0” to enable the clock to the RTC register interface and peripheral clock.
- For RTC software reset, use the RTC CTRL register. See Table 584. The RTC is reset only by initial power-up of the device or when an RTC software reset is applied; it is not initialized by other system resets.
- The RTC provides an interrupt to the NVIC for the RTC_WAKE and RTC_ALARM functions, see Chapter 3 “LPC55S6x/LPC55S2x/LPC552x Nested Vectored Interrupt Controller (NVIC)”.
- To enable the RTC interrupts for waking up from deep-sleep and power-down modes, enable the interrupts using low power API, and also enable in 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, see Table 584.

- If enabled, the RTC and its oscillator continue running in all reduced power modes as long as power is supplied to the device. Therefore, the 32 kHz output is always available to be enabled for syscon clock generation, see Table 37. Once enabled, the 32 kHz clock can be selected for the system clock or be observed through the CLKOUT pin. The 1 Hz output is enabled in the RTC CTRL register (RTC_EN bit). Once the 1 Hz output is enabled, the 1 kHz output for the high-resolution wake-up timer can be enabled in the RTC CTRL register (RTC1kHz_EN bit).
- If the 32 kHz output of the RTC is used by another part of the system, enable it via the EN bit in the RTCOSCCTRL register.



Note: Although the 32 kHz crystal is available, the 32 kHz FRO is used as the clock source in the default configuration.

Fig 107. RTC clocking and block diagram

28.3.1 RTC timers

The RTC contains two counters:

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 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 or deep power-down modes. The high-resolution wake-up timer can be disabled to conserve power if not used.

28.4 General description

28.4.1 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 device is powered up and the RTC remains 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, and deep power-down) an RTC alarm interrupt can also wake up the part to exit the power mode and begin normal operation.

28.4.2 Sub-second counter

The Real Time Clock module include a 15-bit sub-second up-counter which is clocked at a 32 kHz rate. The 32 kHz clock must be enabled prior to using this feature.

The state of this counter may be read via the bus and combined with the main, one-second RTC count for a more precise time reading. The sub-second counter does not contribute to alarm, interrupt, or wake-up generation.

The sub-second counter is in the always-on domain but is disabled whenever the RTC is in reset or the main RTC 1 Hz counter is disabled. It must be independently enabled by setting bit 10 of the RTC Control Register after the main counter in enabled. Once enabled, the counter waits until the start of the next one-second interval and then begins incrementing at a 32 kHz rate. It will roll-over to zero and resume counting at the start of each one-second interval as long as the counter is enabled.

28.4.3 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 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.

28.4.4 General purpose backup registers

The general purpose registers retain data through the deep power-down mode or loss of main power. Only a complete removal of power from the chip or a software reset of the RTC can clear the general purpose registers.

The RTC module offers a set of registers as backup storage, and is reset only on a software reset of the RTC. These registers may be used to store critical data through deep power-down mode.

28.4.5 RTC power

The RTC module and the oscillator that drives it, run directly from device power pins. The RTC module, and the oscillator that drives it, reside in an always-on power domain that retains power through deep-power-down mode. As a result, the RTC timer and sub-second timer can continue running in deep-power-down mode.

28.5 Pin description

Table 582 gives a summary of pins related to the RTC.

Table 582. RTC pin description

Pin	Type	Description
RTCXIN	Input	RTC oscillator input.
RTCXOUT	Output	RTC oscillator output.

28.6 Register description

Reset values pertain to initial power-up of the device or when an RTC software reset is applied (except where noted). This block is not initialized by any other system reset.

Table 583. Register overview: RTC (base address 0x4002 C000)

Name	Access	Offset	Description	SWRESET bit in CTRL = 1	Reset value	Section
CTRL	R/W	0x00	RTC control.	0x1	0x1	28.6.1
MATCH	R/W	0x04	RTC match.	0xFFFF FFFF	0xFFFF FFFF	28.6.2
COUNT	R/W	0x08	RTC counter.	0	0	28.6.3
WAKE	R/W	0x0C	High-resolution/wake-up timer control.	0	0	28.6.4
SUBSEC	RO	0x10	RTC sub-second counter.	0	0	28.6.5
GPREG0	R/W	0x40	General purpose register 0.	0	0	28.6.6
GPREG1	R/W	0x44	General purpose register 1.	0	0	28.6.6
GPREG2	R/W	0x48	General purpose register 2.	0	0	28.6.6
GPREG3	R/W	0x4C	General purpose register 3.	0	0	28.6.6
GPREG4	R/W	0x50	General purpose register 4.	0	0	28.6.6
GPREG5	R/W	0x54	General purpose register 5.	0	0	28.6.6
GPREG6	R/W	0x58	General purpose register 6.	0	0	28.6.6
GPREG7	R/W	0x5C	General purpose register 7.	0	0	28.6.6

28.6.1 RTC CTRL register

This register controls which clock the RTC uses (1 kHz or 1 Hz) and enables the two RTC interrupts to wake up the part from low-power mode.

Table 584. RTC control register (CTRL, offset 0x00)

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 RTC_OSC_PD and RTC_OSC_BYPASS bits in this register. 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.	
1	-	-	Reserved.	1
2	ALARM1HZ		RTC 1 Hz timer alarm flag status.	0
		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.	

Table 584. RTC control register (CTRL, offset 0x00) *…continued*

Bit	Symbol	Value	Description	Reset value
3	WAKE1kHz		RTC 1 kHz (1024 Hz) timer wake-up flag status	0
		0	Run. The RTC 1 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 kHz timer wake-up enable for deep power-down.	0
		0	Disable. A match on the 1 kHz RTC timer will not bring the part out of deep power-down mode.	
		1	Enable. A match on the 1 kHz RTC timer bring the part out of deep power-down mode.	
6	RTC1kHz_EN		RTC 1 kHz clock enable. This bit can be set to 0 to conserve power if the 1 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 kHz RTC timer will not bring the part out of deep power-down mode. Disabling the RTC 1 kHz clock also clears the WAKE1kHz flag.	
		1	Enable. The 1 kHz RTC timer is enabled.	
7	RTC_EN		RTC enable.	0
		0	Disable. The RTC 1 Hz and 1 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 enabled and RTC operation is enabled. This bit must be set 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 kHz clock, set bit 6 in this register.	
8	RTC_OSC_PD		RTC oscillator power-down control.	0
		0	RTC oscillator is powered up. This bit must be cleared in order for the RTC module to function.	
		1	RTC oscillator is powered-down. The RTC oscillator is shut-off to reserve power consumption. RTC operation is disabled.	
9	RTC_OSC_BYPASS		RTC Oscillator Bypass control.	0
		0	The RTC Oscillator operates normally as a crystal oscillator with the crystal connected between the RTC_XTALIN and RTC_XTALOUT pins.	
		1	The RTC Oscillator is in bypass mode. In this mode a clock can be directly input into the RTC_XTALIN pin.	

Table 584. RTC control register (CTRL, offset 0x00) *...continued*

Bit	Symbol	Value	Description	Reset value
10	RTC_SUBSEC_ENA		RTC Sub-second counter control.	0
		0	The sub-second counter is disabled. This bit is cleared by a system-level POR or BOD reset as well as a by the RTC_ENA bit (bit 7 in this register).	
		1	The 32 kHz sub-second counter is enabled. Counting commences on the start of the first one-second interval after this bit is set. Note: This bit can only be set after the RTC_ENA bit (bit 7) is set by a previous write operation.	
31:11	-	-	Reserved. Read value is undefined, only zero should be written.	-

28.6.2 RTC match register

Table 585. RTC match register (MATCH, offset 0x04)

Bit	Symbol	Description	Reset value
31:0	MATVAL	Contains the match value against which the 1 Hz RTC timer will be compared to set the alarm flag RTC_ALARM and generate an alarm interrupt/wake-up if enabled.	0xFFFF FFFF

28.6.3 RTC counter register

Table 586. RTC counter register (COUNT, offset 0x08)

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:** No synchronization is provided to prevent a read of the counter register during a count transition. The suggested method to read a counter is to read the location twice and compare the results. If the values match, the time can be used. If they do not match, then the read should be repeated until two consecutive reads produce the same result. **Remark:** Only write to this register when the RTC_EN bit in the RTC CTRL Register is 0. The counter increments one second after the RTC_EN bit is set.	0

28.6.4 RTC high-resolution/wake-up register

Table 587. RTC high-resolution/wake-up register (WAKE, offset 0x0C)

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. Read value is undefined, only zero should be written.	-

28.6.5 RTC sub-second counter

Table 588. RTC sub-second counter register (SUBSEC, offset 0x10)

Bit	Symbol	Description	Reset value
14:0	SUBSEC	A read reflects the current value of the 32kHz sub-second counter. This counter is cleared whenever the SUBSEC_ENA bit in the RTC_CONTROL register is low. Up-counting at a 32kHz rate commences at the start of the next one-second interval after the SUBSEC_ENA bit is set. This counter must be re-enabled after exiting deep power-down mode or after the main RTC module is disabled and re-enabled. On modules not equipped with a sub-second counter, this register will read-back as all zeroes.	0
31:15	-	Reserved. Read value is undefined, only zero should be written.	-

28.6.6 RTC general purpose backup registers

These register retain contents even during deep power-down mode as long as device power is maintained. They can be used to preserve application data or configuration that will always be available.

Table 589. RTC general purpose registers 0 to 7 (GPREG[0:7], offset 0x40:0x5C)

Bit	Symbol	Description	Reset value
31:0	GPDATA	When implemented, these eight registers can be used to store information through deep power-down mode or loss of main power. Data is retained during deep power-down mode or loss of main power as long as VBAT is supplied.	0

29.1 How to read this chapter

Three system tick timers (SysTick timer) are present in the device. Two inside the CPU0 (Secured and Non-Secured), and one inside the CPU1 (Non-Secured)

Each tick timer has its own calibration provided by the Syscon.

29.2 Features

- Simple 24-bit timer.
- Uses dedicated exception vector.
- Clocked by the CPU clock or by an external clock which can be selected via SYSTICKCLKSElx in Syscon.

29.3 Basic configuration

Configuration of the system tick timer is accomplished as follows:

1. Pins: The system tick timer uses no external pins.
2. Power: The system tick timer is enabled through the SysTick control register.
3. Enable and select the clock source for the SysTick timer in the SYST_CSR register and configure the syscon registers SYSTICKCLKSEL0 and SYSTICKCLKDIV0 (CPU0 Secure and Non-Secure SysTicks), SYSTICKCLKSEL1 and SYSTICKCLKDIV1 (CPU1) if required.

29.4 General description

See Figure 108 for the block diagram of the SysTick timer.

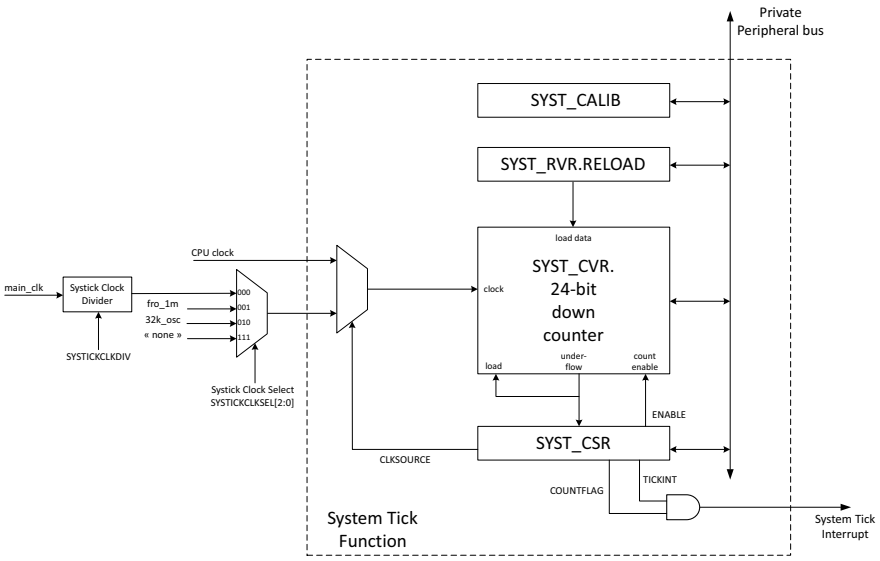


Fig 108. System tick timer block diagram

The SysTick timer is an integral part of the Cortex-M33. 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 CPU, it facilitates porting of software by providing a standard timer that is available on ARM Cortex-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.
- 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 appropriate ARM Cortex User Guide for details.

29.5 Register description

The systick timer registers are located on the private peripheral bus of each CPU (see Figure 4.

Table 590. Register overview: SysTick timer (base address, 0xE000 E000)

Name	Access	Offset	Description	Reset[1] value	Section
SYST_CSR	R/W	0x010	System Timer Control and status register	0	29.5.1
SYST_RVR	R/W	0x014	System Timer Reload value register	0	29.5.2
SYST_CVR	R/W	0x018	System Timer Current value register	0	29.5.3
SYST_CALIB	RO	0x01C	System Timer Calibration value register	0	29.5.4

[1] Reading an writing have specific side effects see detailed register description.

29.5.1 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 CPU.

This register determines the clock source for the system tick timer.

Table 591. SysTick Timer Control and status register (SYST_CSR, offset 0x010)

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 CPU clock is selected. When 0, the external is selected as the reference clock. Refer to syscon YSTICKCLKSEL0 and SYSTICKCLKSEL1. **Remark:** When the output of the main clock divider is selected as the clock source, the CPU clock must be at least 2.5 times faster than the divider output.	0
15:3	-	Reserved. Read value is undefined, only zero should be written.	NA
16	COUNTFLAG	Returns 1 if the SysTick timer counted to 0 since the last read of this register. Cleared automatically when read or when the counter is cleared.	
31:17	-	Reserved. Read value is undefined, only zero should be written.	NA

29.5.2 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 592. System timer reload value register (SYST_RVR, offset 0x014)

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. Read value is undefined, only zero should be written.	NA

29.5.3 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 593. System timer current value register (SYST_CVR, offset 0x018)

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. Read value is undefined, only zero should be written.	NA

29.5.4 System timer calibration value register

The value of the SYST_CALIB register is read-only and is provided by the value of the CPU0STCKCAL (CPU0 Secured), CPU0NSTCKCAL (CPU0 Non-Secured), CPU1TCKCAL (CPU1)registers in the system configuration block. See Table 38.

Table 594. System timer calibration value register (SYST_CALIB, offset 0x01C)

Bit	Symbol	Description	Reset value
23:0	TENMS	Reload value from the SYSTCKCAL register in the SYSCON block. This field is loaded from the CPU0STCKCAL, CPU0NSTCKCAL, CPU1TCKCAL register in Syscon.	0
29:24	-	Reserved. Read value is undefined, only zero should be written.	NA
30	SKEW	Indicates whether the TENMS value will generate a precise 10 millisecond time, or an approximation. This bit is loaded from the CPU0STCKCAL, CPU0NSTCKCA, CPU1TCKCAL register in Syscon. When 0, the value of TENMS is considered to be precise. When 1, the value of TENMS is not considered to be precise.	0
31	NOREF	Indicates whether an external reference clock is available. This bit is loaded from the CPU0STCKCAL, CPU0NSTCKCAL, CPU1TCKCAL register in Syscon. When 0, a separate reference clock is available. When 1, a separate reference clock is not available.	0

29.6 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 or from an external clock, see Figure 4. 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.
The clock used for the SysTick timer maybe be selected as the output of the SYSTICK

clock divider. Therefore, its frequency depends on Main_Clk frequency and the clock

divider settings driven from Syscon registers SYSTICKCLKDIV0 (CPU0) and SYSTICKCLKDIV1 (CPU1).The Sys Tick register, SYST_CALIB, is a read only register. The field TENMS can be used to indicate the number of ticks needed for a 10ms period. To set this value, write the required value to SYSCON CPU0STCKCAL, CPU0NSTCKCAL, CPU1TCKCAL register. This value is based on the clock settings previously described.

The two further fields in SYST_CALIB are also driven from the CPU0STCKCAL, CPU0NSTCKCAL, CPU1TCKCAL registers, using the SKEW and NOREF fields.

29.7 Example timer calculations

To use the system tick timer, do the following:

1. Program the SYST_RVR register with the reload value calculated as shown below 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. The following examples illustrate selecting SysTick timer reload values for different system configurations. All of the examples calculate an interrupt interval of 10 milliseconds, as the SysTick timer is intended to be used, and there are no rounding errors.

System tick timer clock = 12 MHz, CPU clock = 48 MHz

Program the SYST_CSR register with the value 0x3 which selects the external clock source. See Section 4.5.46 “SYSTICK clock divider register 0” and Section 4.5.26 “System Tick Timer for CPU0 source select”.

Use DIV of the SYSTICKCLKDIV0 (CPU0) and SYSTICKCLKDIV1 (CPU1) setting.

SYST_RVR = (system tick timer clock frequency x 10 ms) -1 = (12 MHz x 10 ms) -1 = 120000 - 1 = 119999 = 0x0001 D4BF

CPU clock = 12 MHz

Program the SYST_CSR register with the value 0x7 which selects the CPU clock as the clock source and enables the SysTick timer and the SysTick timer interrupt.

SYST_RVR = (CPU clock frequency x 10 ms) - 1 = (12 MHz x 10 ms) - 1 = 120000 - 1 = 119999 = 0x0001 D4BF

30.1 How to read this chapter

The watchdog timer is available on all LPC55S6x/LPC55S2x/LPC552x devices.

30.2 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 four 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 / watchdog feed sequence can optionally be protected such that it can only be performed after the “warning interrupt” time is reached.
- Flag to indicate Watchdog reset.
- The watchdog clock (WDCLK) is generated from always on FRO_1MHz clock, see Figure 109 that can be divided by WDT clock divider register value in SYSCON module. See Table 105. The accuracy of this clock is limited to +/- 15% over temperature, voltage, and silicon processing variations. To determine the actual watchdog frequency, use the frequency measure block. See Chapter 11 “LPC55S6x/LPC55S2x/LPC552x Analog control”
- The watchdog timer can be configured to run in deep-sleep mode.
- Debug mode.

30.3 Basic configuration

Configuration of the WWDT is accomplished as follows:

- Configure WDTCLKDIV register. See Table 105. Release the reset, disable HALT bit and program DIV[5:0].
- Enable the register interface (WWDT bus clock): set the WWDT bit in the AHBCLKCTRL0 register, see Table 55.
- For waking up from a WWDT interrupt, enable the watchdog interrupt for wake-up using low power API.

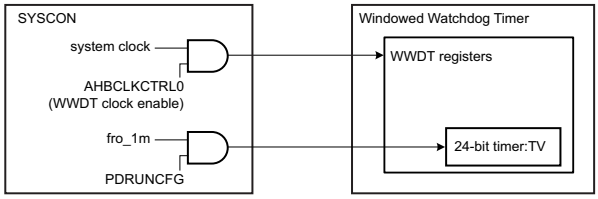


Fig 109. WWDT clocking

30.4 Pin description

The WWDT has no external pins.

30.5 General description

The purpose of the watchdog timer is to reset or interrupt the micro-controller within a programmable time if it enters an erroneous state. When enabled, a watchdog reset is generated if the user program fails to feed (reload) the watchdog within a predetermined amount of time.

When a watchdog window is programmed, an early watchdog feed is also treated as a watchdog event. This allows preventing situations where a system failure may still feed the watchdog. For example, application code could be stuck in an interrupt service that contains a watchdog feed. Setting the window such that this would result in an early feed will generate a watchdog event, allowing for system recovery.

The watchdog consists of a fixed (divide by 4) pre-scaler and a 24-bit counter which decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum Watchdog interval is ($T_{WDCLK} \times 256 \times 4$) and the maximum watchdog interval is ($T_{WDCLK} \times 2^{24} \times 4$) in multiples of ($T_{WDCLK} \times 4$). The watchdog should be used in the following manner:

- Enable and configure the watchdog clock as described in Section 30.3 “Basic configuration”.
- 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.
- Set the watchdog timer update mode (WDPROTECT) in the MOD register after a delay of three WDCLK clock cycles.

- 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 CPUs 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 is no longer greater than the value defined by the WARNINT register.

30.5.1 Block diagram

The block diagram of the watchdog is shown in the Figure 110. The synchronization logic (APB bus clock to WDCLK) is not shown in the block diagram.

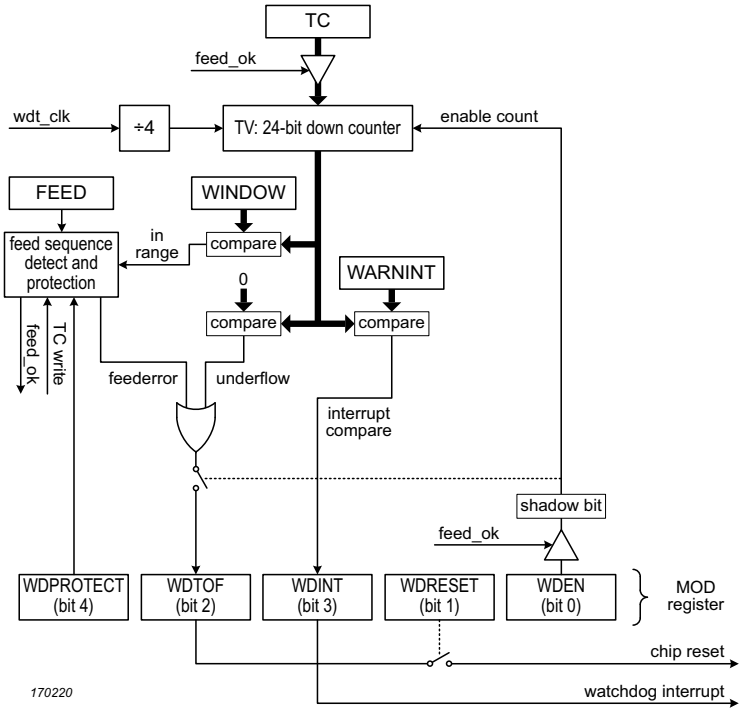


Fig 110. Windowed watchdog timer block diagram

30.5.2 Clocking and power control

The watchdog timer block uses two clocks: APB bus clock and WDCLK. The APB bus clock is used for the APB accesses to the watchdog registers and is derived from the system clock, see Figure 110. The WDCLK is used for the watchdog timer counting and is derived from the FRO_1MHz that can be divided.

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 the APB bus clock, so that the CPU can read the TV 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.

30.5.3 Using the WWDT lock feature

The WWDT supports a lock feature which can be enabled to ensure that the WWDT is running at all times:

- Performing the WWDT reload / WWDT feed sequence.

30.5.3.1 Changing the WWDT reload value

If bit 4 is set in the WWDT MOD register, the watchdog reload / watchdog feed sequence can be performed only after the watchdog timer is below the value of WDWARNING and WDWINDOW.

The reload overwrite lock mechanism can only be disabled by a reset of any type.

30.6 Register description

The watchdog timer contains the registers shown in Table 595.

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

Table 595. Register overview: watchdog timer (base address 0x4000 C000)

Name	Access	Offset	Description	Reset value	Section
MOD	R/W	0x000	Watchdog mode. This register contains the basic mode and status of the watchdog timer.	0	30.6.1
TC	R/W	0x004	Watchdog timer constant. This 24-bit register determines the time-out value.	0xFF	30.6.2
FEED	WO	0x008	Watchdog feed sequence. Writing 0xAA followed by 0x55 to this register reloads the watchdog timer with the value contained in TC.	NA	30.6.3
TV	RO	0x00C	Watchdog timer value. This 24-bit register reads out the current value of the watchdog timer.	0xFF	30.6.4
-	-	0x010	Reserved	-	-
WARNINT	R/W	0x014	Watchdog warning interrupt compare value.	0	30.6.5
WINDOW	R/W	0x018	Watchdog window compare value.	0xFF FFFF	30.6.6

30.6.1 Watchdog mode register

The WDMOD register controls the operation of the watchdog. Note that a watchdog feed must be performed before any changes to the WDMOD register take effect.

Table 596. Watchdog mode register (MOD, offset 0x000)

Bit	Symbol	Value	Description	Reset value
0	WDEN		Watchdog enable bit. Once this bit is set to one and a watchdog feed is performed, the watchdog timer will run permanently.	0
		0	Stop. The watchdog timer is stopped.	
		1	Run. The watchdog timer is running.	
1	WDRESET		Watchdog reset enable bit. Once this bit has been written with a 1 it cannot be re-written with a 0.	0
		0	Interrupt. A watchdog time-out will not cause a chip reset.	
		1	Reset. A watchdog time-out will cause a chip reset.	
2	WDTOF	-	Watchdog time-out flag. Set when the watchdog timer times out, by a feed error, or by events associated with WDPROTECT. Cleared by software writing a 0 to this bit position. Causes a chip reset if WDRESET = 1.	0 [1]
3	WDINT	-	Warning interrupt flag. Set when the timer is at or below the value in WDWARNINT. Cleared by software writing a 1 to this bit position. Note that this bit cannot be cleared while the WARNINT value is equal to the value of the TV register. This can occur if the value of WARNINT is 0 and the WDRESET bit is 0 when TV decrements to 0.	0

Table 596. Watchdog mode register (MOD, offset 0x000)

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	Flexible. The watchdog reload / watchdog feed sequence can be performed when the watchdog timer is below the value of WDWINDOW.	
		1	Threshold. The watchdog reload / watchdog feed sequence can be performed only after the watchdog timer is below the value of WDWARNING and WDWINDOW.	
31:5	-	-	Reserved. Read value is undefined, only zero should be written.	-

[1] Only an external or power-on reset has this effect.

Once the **WDEN**, **WDPROTECT**, or **WDRESET** bits are set they can not be cleared by software. All three bits 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 is no longer greater than 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 sleep and deep-sleep low power modes, a watchdog reset or interrupt can occur when the watchdog is running and has an operating clock source. The watchdog clock can be configured to keep running in sleep and deep-sleep modes.

If a watchdog interrupt occurs in sleep or deep-sleep mode, has been enabled using the POWER_EnterDeepSleep() API, the device will wake up.

See the following registers:

Table 597. 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 micro-controller. A watchdog feed prior to reaching the value of WDWINDOW will also cause a watchdog reset.

30.6.2 Watchdog timer constant register

The TC register determines the time-out value. Every time a feed sequence occurs the value in the TC is loaded into the watchdog timer. The TC resets to 0x00 00FF. Writing a value below 0xFF will cause 0x00 00FF to be loaded into the TC. Thus the minimum time-out interval is $T_{WDCLK} \times 256 \times 4$.

If the WDPROTECT bit in WDMOD = 1, an attempt to perform the watchdog reload / watchdog feed sequence before the watchdog timer is below the values of WDWARNINT and WDWINDOW will cause a watchdog feed error and set the WDTOF flag.

Table 598. Watchdog timer constant register (TC, offset 0x04)

Bit	Symbol	Description	Reset value
23:0	COUNT	Watchdog time-out value.	0x00 00FF
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

30.6.3 Watchdog feed register

Writing 0xAA followed by 0x55 to this register will reload the watchdog timer with the TC 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 APB bus clock following an incorrect access to a watchdog register during a feed sequence.

It is good practice to disable interrupts around a feed sequence, if the application is such that an interrupt might result in rescheduling processor control away from the current task in the middle of the feed, and then lead to some other access to the WDT before control is returned to the interrupted task.

Table 599. Watchdog feed register (FEED, offset 0x08)

Bit	Symbol	Description	Reset value
7:0	FEED	Feed value should be 0xAA followed by 0x55.	NA
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

30.6.4 Watchdog timer value register

The TV 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 APB bus clock cycles, so the value of TV is older than the actual value of the timer when it's being read by the CPU.

Table 600. Watchdog timer value register (TV, offset 0x0C)

Bit	Symbol	Description	Reset value
23:0	COUNT	Counter timer value.	0x00 00FF
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

30.6.5 Watchdog timer warning interrupt register

The WDWARNINT register determines the watchdog timer counter value that will generate a watchdog interrupt. When the watchdog timer counter is no longer greater than the value defined by WARNINT, an interrupt will be generated after the subsequent WDCLK.

A match of the watchdog timer counter to WARNINT occurs when the bottom 10 bits of the counter have the same value as the 10 bits of WARNINT, and the remaining upper bits of the counter are all 0. This gives a maximum time of 1,023 watchdog timer counts (4,096 watchdog clocks) for the interrupt to occur prior to a watchdog event. If WARNINT is 0, the interrupt will occur at the same time as the watchdog event.

Table 601. Watchdog timer warning interrupt register (WARNINT, offset 0x14)

Bit	Symbol	Description	Reset value
9:0	WARNINT	Watchdog warning interrupt compare value.	0
31:10	-	Reserved, only zero should be written.	-

30.6.6 Watchdog timer window register

The WINDOW register determines the highest TV value allowed when a watchdog feed is performed. If a feed sequence occurs when TV is greater than the value in WINDOW, a watchdog event will occur.

WINDOW resets to the maximum possible TV value, so windowing is not in effect.

Table 602. Watchdog timer window register (WINDOW, offset 0x18)

Bit	Symbol	Description	Reset value
23:0	WINDOW	Watchdog window value.	0xFF FFFF
31:24	-	Reserved, only zero should be written.	-

30.7 Functional description

The following figures illustrate several aspects of watchdog timer operation.

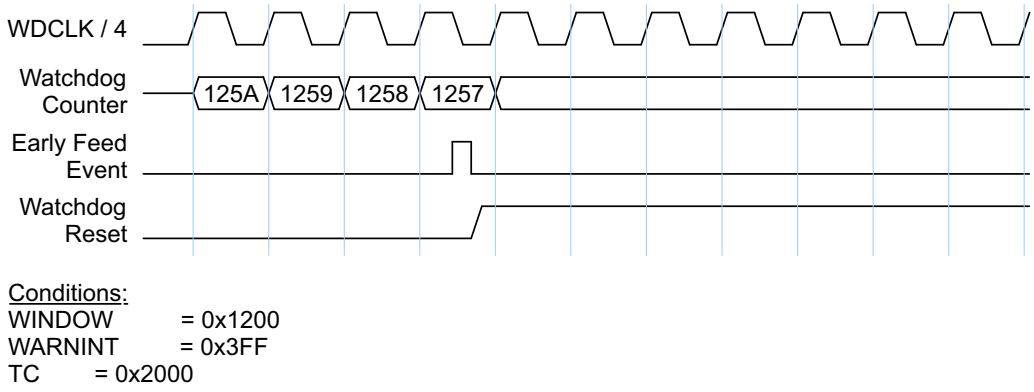


Fig 111. Early watchdog feed with windowed mode enabled

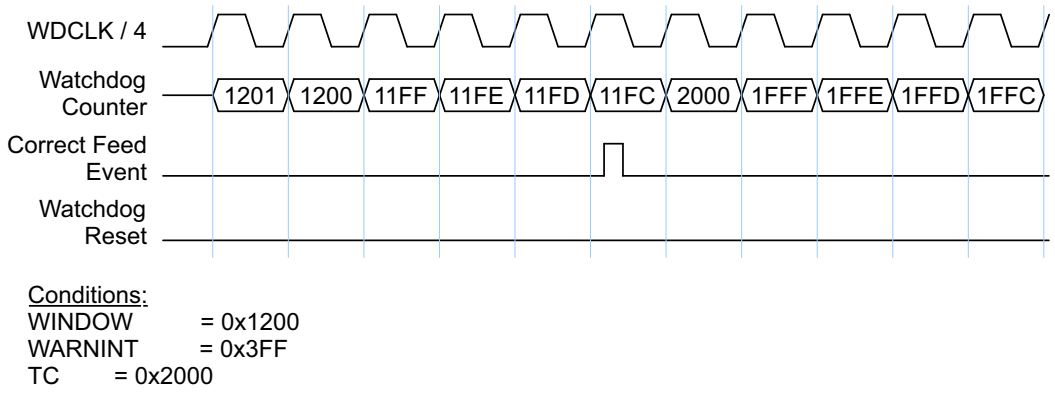


Fig 112. Correct watchdog feed with windowed mode enabled

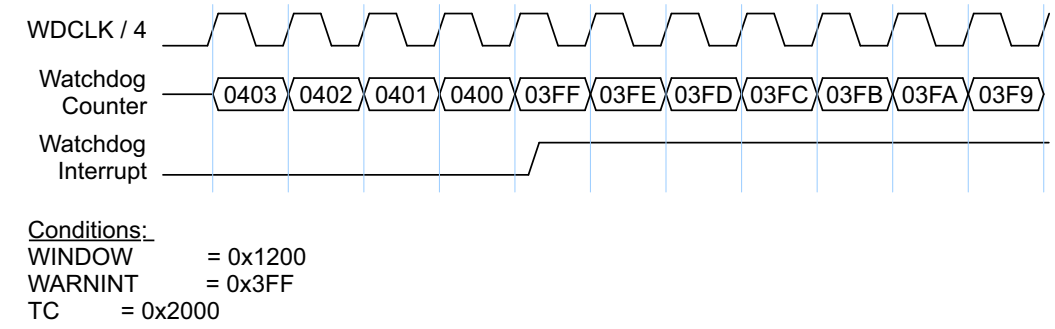


Fig 113. Watchdog warning interrupt

31.1 How to read this chapter

The OS Event timer is available on all LPC55S6x/LPC55S2x/LPC552x devices.

31.2 Features

- Central 42-bit, free-running gray-code event/timestamp timer.
- Match registers compared to the main counter to generate an interrupt and/or wake-up event.
- Capture registers triggered by CPU command, readable via the APB bus.
- APB interface for register access.
- IRQ and wake-up.
- Reads of gray-encoded timers are accomplished with no synchronization latency.
- Located in the always-on domain, so that it can wake up the device from all low power modes, including deep power-down.

31.3 Basic configuration

Configure the OS Event timer as follows

- Use the AHBCLKCTRL1 register in SYSCON, to enable the clock to the OS Event timer register interface and use the OSTIMER register, to enable the clock 32k peripheral clock. To enable FRO/XTAL 32 kHz output clock use RTCOSC32K register in PMC.
- Use the PRESETCTRL1 register in SYSCON, to clear the reset to the OS Event timer. Clear OS Event Timer Interrupt flag. Read the event timer until it increments. Enable Systems Interrupts in the OS Event Timer. Clear the OS Event Timer Interrupt flag. Enable OS Event Timer interrupt in the NVIC
- For OS Event timer software reset use the OSTIMER register.
- OS Event timer provides an interrupt to the NVIC.
- This module is placed in Always-ON domain, and hence can be running in all low-power modes including deep power-down. It can be a wake-up source in deep power-down mode.
- To enable the OS Event timer interrupt for waking up from deep-sleep and power-down modes, in deep power-down mode, use the relevant low power API.
- To enable the OS Event timer interrupt for waking up from deep power-down, enable the wake-up in the OSTIMER register in PMC.

31.4 Pin description

The OS Event timer is not associated with any device pins.

31.5 General description

The OS Event timer is comprised of one central 42-bit timer (“EVTimer”), and separate 42-bit match and capture registers and a maskable IRQ/wake-up request. Figure 114 shows a conceptual view of the OS Event timer.

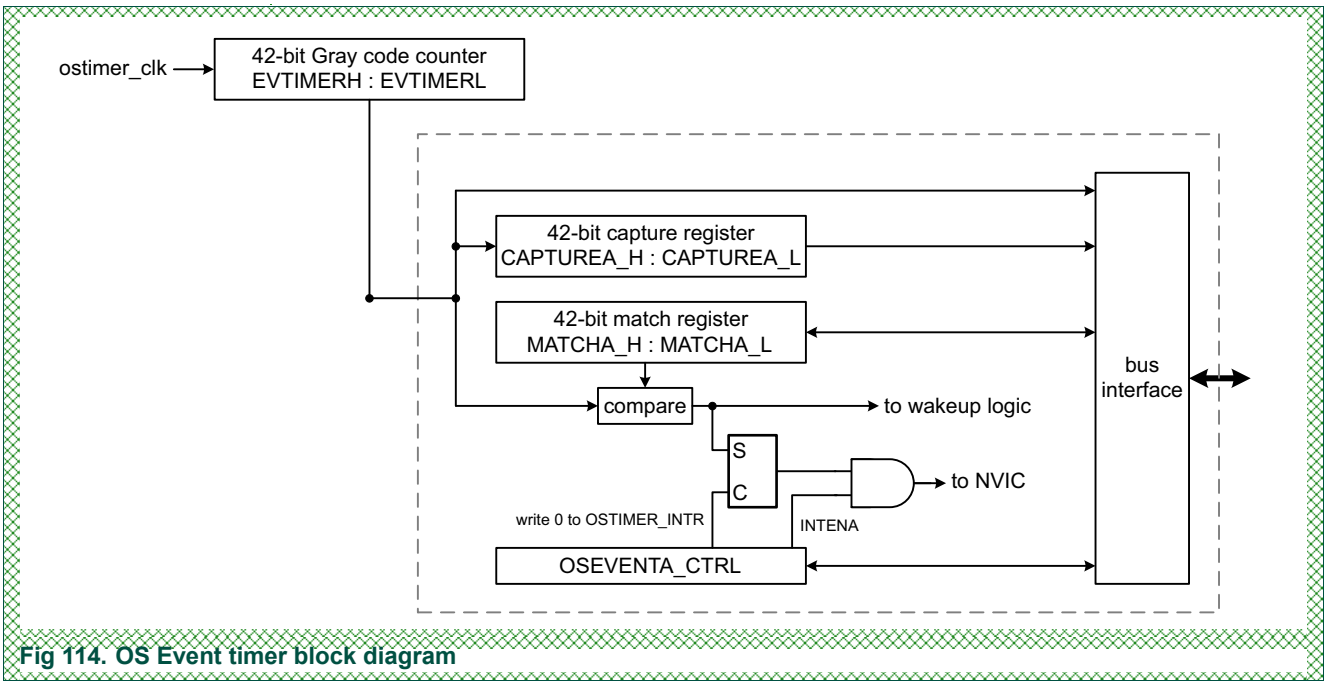


Fig 114. OS Event timer block diagram

31.5.1 Central Event/timestamp timer

The 42-bit central EVTimer is initialized on a full-system POR and then counts up continuously. The typical clock for this timer is a 32 kHz clock.

The central EVTimer is implemented as a gray-coded counter and can be read from capture registers.

31.5.2 Match, capture, and interrupt generation

The timer includes capture registers, match registers, and a control register.

Capture registers

42-bits of capture values are available in the Capture_L and Capture_H registers. A capture command issued by the CPU (the capture command is issued by an Arm "Set Event" intrinsic instruction or in CMSIS C-coding by the function "__SEV();") causes the current value of the main EVTimer to be stored in the capture registers. The capture registers are clocked by the same clock as the associated CPU. Use of gray encoding eliminates issues associated with the asynchronous transfer from the EVTimer to the capture registers.

Match registers and interrupt request

42-bits of match values are available in the Match_H and Match_L registers. The EVTimer output is compared against this combined value for interrupt/wake-up generation. A match to this register pair will set a flag which can be enabled to generate the interrupt/wake-up request. The value written to the match register pair must also be specified in gray code.

Writes to the match registers are stored in domain bus clock based shadow registers. They are automatically transferred to the actual match registers following a write to the Match_H register. For this reason, the Match_L register must always be written first, followed by the write to the Match_H register.

Reads of the match registers will reflect the value in the bus clock based shadow registers so that a read immediately following a write will always reflect the value just written.

31.6 Register description

Table 603. Register overview: OS Event timer (base address = 0x4002D000)

Name	Access	Offset	Description	Reset value	Section
EVTIMERL	RO	0x0	Central EVTIMER low register.	0x0	31.6.1
EVTIMERH	RO	0x4	Central EVTIMER high register.	0x0	31.6.2
CAPTURE_L	RO	0x8	Capture low register.	0x0	31.6.3
CAPTURE_H	RO	0xC	Capture high register.	0x0	31.6.4
MATCH_L	RW	0x10	Match low register.	0xFFFFFFFF	31.6.5
MATCH_H	RW	0x14	Match high register.	0xFFFFFFFF	31.6.6
OSEVENT_CTRL	RW	0x1C	OS_EVENT TIMER control register.	0x0	31.6.7

31.6.1 Central EVTIMER low register (EVTIMERL)

This register resets only on POR or a software reset.

Table 604. EVTIMER low register (EVTIMERL, offset = 0x0)

Bit	Symbol	Description	Reset value
31:0	EVTIMER_COUNT_VALUE	A read reflects the current value of the lower 32 bits of the EVTIMER. Note: There is only one EVTIMER, readable from all domains.	0x0

31.6.2 Central EVTIMER high register (EVTIMERH)

This register resets only on POR or a software reset.

Table 605. EVTIMER high register (EVTIMERH, offset = 0x4)

Bit	Symbol	Description	Reset value
9:0	EVTIMER_COUNT_VALUE	A read reflects the current value of the upper 10 bits of the 42-bit EVTIMER value. Note: There is only one EVTMER readable from all domains.	0x0
31:10	-	Reserved.	

31.6.3 Capture low register (CATURE_L)

This register resets only on system reset. Not effected by a software reset.

Table 606. Capture low register for CPUn (CAPTURE_L, offset = 0x8)

Bit	Symbol	Description	Reset value
31:0	CAPTURE_VALUE	A read reflects the value of the lower 32 bits of the central EVTIMER at the time the last capture signal was generated by the CPU (using CMSIS C function "__SEV();").	0x0

31.6.4 Capture high register (CAPTURE_H)

This register resets only on system reset. Not effected by a software reset.

Table 607. Capture high register for CPUn (CAPTURE_H, offset = 0xC)

Bit	Symbol	Description	Reset value
9:0	CAPTURE_VALUE	A read reflects the value of the upper 10 bits of the central 42-bit EVTIMER at the time the last capture signal was generated by the CPU (using CMSIS C function "__SEV();").	0x0
31:10	-	Reserved.	

31.6.5 Match low register (MATCH_L)

This register resets only on system reset. Not effected by a software reset.

Table 608. Match low register for CPUn (MATCH_L, offset = 0x10)

Bit	Symbol	Description	Reset value
31:0	MATCH_VALUE	The value written to the MATCH (L/H) register pair is compared against the central EVTIMER. When a match occurs, an interrupt request is generated if enabled.	0xFFF FFFFF

31.6.6 Match high register (MATCH_H)

This register resets only on system reset. Not effected by a software reset.

Table 609. Match high register for CPUn (MATCH_H, offset = 0x14)

Bit	Symbol	Description	Reset value
9:0	MATCH_VALUE	The value written (upper 10 bits) to the MATCH (L/H) register pair is compared against the central EVTIMER. When a match occurs, an interrupt request is generated if enabled.	0xFFF FFFFF
31:10	-	Reserved.	

31.6.7 OS_EVENT control register (OSEVENTn_CTRL)

This register resets only on system reset or a software reset.

Table 610. OS_EVENT TIMER control register for CPUn (OSEVENTn_CTRL, offset = 0x1C)

Bit	Symbol	Description	Reset value
0	OSTIMER_INTRFLAG	This bit is set when a match occurs between the central 42-bit EVTIMER and the value programmed in the match-register pair. This bit is cleared by writing a '1'. Writes to clear this bit are asynchronous. It should be done before a new match value is written into the MATCH_L/H registers.	0x0
1	OSTIMER_INTENA	When this bit is '1' an interrupt/wake-up request to the domain processor will be asserted when the OSTIMER_INTR FLAG is set. When this bit is '0', interrupt/wake-up requests due to the OSTIMER_INTR flag are blocked.	0x0
2	MATCH_WR__RDY	This bit will be low when it is safe to write to reload the Match Registers. In typical applications it should not be necessary to test this bit. [1]	0x0
31:3	-	Reserved.	undefined

[1]The 42-bit MATCH Register value is transferred from a pair of shadow registers to the active Match registers following the write to the Match_H register. A second write to the Match Registers should not be initiated until after this transfer completes, as indicated by this status bit returning low. There is no need to test this status bit if an interrupt due to the first match value has already occurred, or if it is certain via some other means that the required period of time (3 bus clocks) has elapsed.

32.1 How to read this chapter

Multiple Flexcomm Interfaces are available on all LPC55S6x/LPC55S2x/LPC552x devices.

32.2 Introduction

Each Flexcomm Interface provides one peripheral function from a choice of several, chosen by the user. This chapter describes the overall Flexcomm Interface and how to choose peripheral functions. Details of the different peripherals are found in separate chapters for each type.

32.3 Features

Each Flexcomm Interface provides a choice of peripheral functions, one of which must be chosen by the user before the function can be configured and used.

- USART with asynchronous operation or synchronous master or slave operation.
- SPI master or slave, with up to four slave selects.
- I^2C, including separate master, slave, and monitor functions.
- I^2S master or slave. Supports single I^2S channel.
- Data for USART, SPI, and I^2S traffic uses the Flexcomm Interface FIFO. The I^2C function does not use the FIFO.

32.4 Basic configuration

The Flexcomm Interface is configured as follows:

1. Peripheral clock: Make sure that the related Flexcomm Interface is enabled in the AHBCLKCTRL1 register, see Section 4.5.18 “AHB clock control 1”.
2. Flexcomm Interface clock: Select a clock source for the related Flexcomm Interface. Options are shown in Figure 4. Also, see Section 4.5.41 “Flexcomm Interface clock source select registers”.
3. Select the required Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface, see Section 32.7.1 “Peripheral Select and Flexcomm Interface ID register”.
4. See specific peripheral chapters for information on configuring those peripherals: Chapter 34 “LPC55S6x/LPC55S2x/LPC552x USARTs”, Chapter 35 “LPC55S6x/LPC55S2x/LPC552x Serial Peripheral Interfaces”, Chapter 33 “LPC55S6x/LPC55S2x/LPC552x I^2C-bus Interfaces”, and Chapter 37 “LPC55S6x/LPC55S2x/LPC552x I^2S interface”.

   Remark: The Flexcomm interface function clock frequency (at the output of the FRG associated with the Flexcomm) shall not be above 100 MHz.

32.5 Architecture

The overall structure of one Flexcomm Interface is shown in Figure 115.

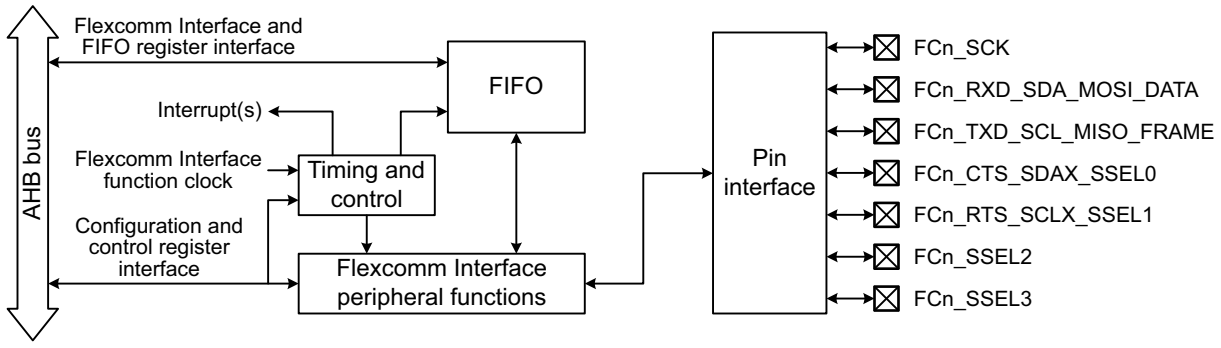


Fig 115. Flexcomm Interface block diagram

32.5.1 Function Summary

LPC55S6x/LPC55S2x/LPC552x devices include Flexcomm Interfaces and functions as shown in Table 611. Specific part numbers and package variations may include a subset of this list.

Table 611. Flexcomm Interface base addresses and functions

Flexcomm Interface number	Base address	USART Chapter 34	SPI Chapter 35	I²C Chapter 33	I²S Chapter 37
0	0x4008 6000	Yes	Yes	Yes	Yes, 1 channel pair.
1	0x4008 7000	Yes	Yes	Yes	Yes, 1 channel pair.
2	0x4008 8000	Yes	Yes	Yes	Yes, 1 channel pair.
3	0x4008 9000	Yes	Yes	Yes	Yes, 1 channel pair.
4	0x4008 A000	Yes	Yes	Yes	Yes, 1 channel pair.
5	0x4009 6000	Yes	Yes	Yes	Yes, 1 channel pair.
6	0x4009 7000	Yes	Yes	Yes	Yes, 1 channel pair.
7	0x4009 8000	Yes	Yes	Yes	Yes, 1 channel pair.

Remark: The FlexComm Interface10 has a different clock source selection than FlexComm Interface 0 to FlexComm Interface 7. See Chapter 4 "LPC55S6x/LPC55S2x/LPC552x SYSCON".

32.5.2 Choosing a peripheral function

A specific peripheral function, from among those supported by a particular Flexcomm Interface, is selected by software writing to the PSELID register. Reading the PSELID register provides information on which peripheral functions are available on that Flexcomm Interface.

Once a specific peripheral function has been selected, the PID register will supply an identifier for the selected peripheral. Software may use this information to confirm the selection before proceeding.

32.5.3 FIFO usage

Refer to the chapter for a specific peripheral function for information on how the FIFO is used, see Table 611.

32.5.4 DMA

The Flexcomm Interface generates DMA requests if desired, based on a selectable FIFO level. See the chapter for a specific peripheral function for information on how the FIFO is used, Table 611.

32.5.5 AHB bus access

Generally, the bus interface to the registers contained in the Flexcomm Interface (including its serial peripheral functions) support only word writes. Byte and halfword writes should not be used. An exception is that the FIFOWR register, when the Flexcomm Interface is configured for use as an SPI interface, also allows byte and halfword writes. This allows support for control information embedded in DMA buffers, for example. See Section 34.6.16 “FIFO write data register” for more information.

32.6 Pin description

Each Flexcomm Interface allows up to 7 pin connections. Specific uses of a Flexcomm Interface typically do not use all of these, and some Flexcomm Interface instances may not provide a means to connect all functions to device pins. Pin usage for a specific peripheral function is described in the chapter for that peripheral.

Table 612. Flexcomm Interface pin description

Pin	Type	Description
SCK	I/O	Clock input or output for the USART function in synchronous modes.
	I/O	Clock input or output for the SPI function.
	I/O	Clock input or output for the I^2S function.
RXD_SDA_MOSI or RXD_SDA_MOSI_DATA	Input	Receive data input for the USART function.
	I/O	SDA (data) input/output for the I^2C function.
	I/O	Master data output/slave data input for the SPI function.
	I/O	Data input or output for the I^2S function.
TXD_SCL_MISO or TXD_SCL_MISO_WS	Output	Transmit data output for the USART function.
	I/O	SCL input/output for the I^2C function.
	I/O	Master data input/slave data output for the SPI function.
	I/O	WS (also known as LRCLK) input or output for the I^2S function.
CTS_SDA_SSEL0	Input	Clear to send input for the USART function.
	I/O	SDA (data) input/output for the I^2C function.
	I/O	Slave select 0 input or output for the SPI function.
RTS_SCL_SSEL1	Output	Request to send output for the USART function.
	I/O	SCL (clock) input/output for the I^2C function.
	I/O	Slave select 1 input or output for the SPI function.
SSEL2	I/O	Slave select 2 input or output for the SPI function.
SSEL3	I/O	Slave select 3 input or output for the SPI function.

32.7 Register description

Each Flexcomm Interface contains registers that are related to configuring the Flexcomm Interface to do a specific peripheral function and other registers related to peripheral FIFOs and data access. The latter depend somewhat on the chosen peripheral functions and are described in the chapters for each specific function (USART, SPI, I^2C, and I^2S if present in a specific Flexcomm Interface). The Flexcomm Interface registers that identify and configure the Flexcomm Interface are shown in Table 611.

The base addresses of all Flexcomm Interfaces may be found in Table 611.

Table 613. Register map for the first channel pair within one Flexcomm Interface

Name	Access	Offset	Description	Reset Value [1]	Section
PSELID	R/W	0xFF8	Peripheral Select and Flexcomm Interface ID register.	0	32.7.1
PID	RO	0xFFC	Peripheral identification register.	0	32.7.2

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

32.7.1 Peripheral Select and Flexcomm Interface ID register

The PSELID register identifies the Flexcomm Interface and provides information about which peripheral functions are supported by each Flexcomm Interface. It also provides the means to select one peripheral function for each Flexcomm Interface.

Table 614. Peripheral Select and Flexcomm Interface ID register (PSELID - offset 0xFF8)

Bit	Symbol	Value	Description	Reset Value
2:0	PERSEL		Peripheral Select. This field is writable by software.	0
		0x0	No peripheral selected.	
		0x1	USART function selected.	
		0x2	SPI function selected.	
		0x3	I^2C function selected.	
		0x4	I^2S transmit function selected.	
		0x5	I^2S receive function selected.	
		0x6	Reserved.	
		0x7	Reserved.	
3	LOCK		Lock the peripheral select. This field is writable by software.	0
		0	Peripheral select can be changed by software.	
		1	Peripheral select is locked and cannot be changed until this Flexcomm Interface or the entire device is reset.	
4	USARTPRESENT		USART present indicator. This field is Read-only.	0
		0	This Flexcomm Interface does not include the USART function.	
		1	This Flexcomm Interface includes the USART function.	
5	SPIPRESENT		SPI present indicator. This field is Read-only.	0
		0	This Flexcomm Interface does not include the SPI function.	
		1	This Flexcomm Interface includes the SPI function.	
6	I2CPRESENT		I^2C present indicator. This field is Read-only.	0
		0	This Flexcomm Interface does not include the I^2C function.	
		1	This Flexcomm Interface includes the I^2C function.	

Table 614. Peripheral Select and Flexcomm Interface ID register (PSELID - offset 0xFF8) *…continued*

Bit	Symbol	Value	Description	Reset Value
7	I2SPRESENT		I^2S present indicator. This field is Read-only.	0
		0	This Flexcomm Interface does not include the I^2S function.	
		1	This Flexcomm Interface includes the I^2S function.	
11:8	-		Reserved. Read value is undefined, only zero should be written.	NA
31:12	ID		Flexcomm Interface ID.	0x00102

32.7.2 Peripheral identification register

This register is read-only and will read as 0 until a specific Flexcomm Interface function is selected via the PID register. Once the Flexcomm Interface is configured for a function, this register confirms the selection by returning the module ID for that function, and identifies the revision of that function. A software driver could make use of this information register to implement module type or revision specific behavior.

Table 615. Peripheral identification register (PID - offset 0xFFC)

Bit	Symbol	Description	Reset Value
7:0	-	-	0
11:8	Minor_Rev	Minor revision of module implementation.	See specific device chapter
15:12	Major_Rev	Major revision of module implementation.	See specific device chapter
31:16	ID	Module identifier for the selected function.	See specific device chapter

33.1 How to read this chapter

I2C-bus functions are available on all LPC55S6x/LPC55S2x/LPC552x devices as a selectable function in each Flexcomm Interface. Up to 8 Flexcomm Interfaces are available.

33.2 Features

- Independent master, slave, and monitor functions
- Bus speeds supports
  - Standard-mode, up to 100kbits/s
  - Fast-mode, up to 400 kbits/s
  - Fast-mode Plus, up to 1 Mbits/s (on specific I2C pins).
  - High-speed mode, 3.4 Mbits/s as a slave only (on specific I2C pins).
- Supports both Multi-master and Multi-master with slave functions.
- Multiple I2C slave address supported in hardware
- One slave address can be selectively qualified with a bit mask or an address range in order to respond to multiple I2C bus addresses.
- 10-bit addressing supported with software assist.
- Supports System Management Bus (SMBus).
- Separate DMA requests for master, slave, and monitor functions.
- No chip clocks are required in order to receive and compare an address as a slave, so this event can wake-up the device from deep-sleep mode.
- Automatic modes optionally allow less software overhead for some use cases.

33.3 Pin description

The I2C pins are fixed-pin functions and enabled through IOCON. See the IOCON settings in Table 616 and in Chapter 15 “LPC55S6x/LPC55S2x/LPC552x I/O Pin Configuration (IOCON)”.

Table 616. I2C-bus pin description

Function	Type	Pin name used in data sheet	Description
SCL	I/O	FCn_TXD_SCL_MISO_WS or FCn_RTS_SCL_SSEL1	I2C serial clock.
SDA	I/O	FCn_RXD_SDA_MOSI_DATA or FCn_CTS_SDA_SSEL0	I2C serial data.

33.4 Basic configuration

Configure the I2C and related clocks as follows:

- If needed, use the PRESETCTRL1 or PRESETCTRL2 register, see Section 4.5.8 “Peripheral reset control 1” and Section 4.5.9 “Peripheral reset control 2” to reset the Flexcomm Interface that is about to have a specific peripheral function selected.
- Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface (flexcom section ref). Note that any selection that has been made will be cleared if the Flexcomm Interface itself is reset via the PRESETCTRL1 or PRESETCTRL2 register.
- Configure the I2C for the desired functions:
  - In Section 4.5.18 “AHB clock control 1”, set the appropriate bit for the related Flexcomm Interface in order to enable the clock to the register interface.
  - Enable or disable the related Flexcomm Interface interrupt in the NVIC, see Table 8.
  - Configure the related Flexcomm Interface pin functions via IOCON, see Chapter IOCON. Configure the I2C clock and data rate. This includes the CLKDIV register for both master and slave modes, and MSTTIME for master mode. See Section 33.6.6 “Time-out value register” and Section 33.7.2 “Bus rates and timing considerations”.

    Remark: The Flexcomm interface function clock frequency (at the output of the FRG associated with the Flexcomm) shall not be above **100 MHz.**

Remark: While the I2C function is incorporated into the Flexcomm Interface, it does not make use of the Flexcomm Interface FIFO.

33.4.1 I2C transmit/receive in master mode

In this example, Flexcomm Interface 1 is configured as an I2C master. The master sends 8 bits to the slave and then receives 8 bits from the slave.

If specialized I2C pins are used (PIO0_13 through PIO0_14), the pins should be configured as required for the I2C-bus mode that will be used (SM, FM, FM+, HS) via the IOCON block. If these or standard pins are used, they should be configured as described in IOCON section.

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

Configure the I2C bit rate:

- Select a source for the Flexcomm Interface 1 clock that will allow for the desired I2C-bus rate. Divide the clock as needed, see Table 629.
- Further divide the source clock if needed using the CLKDIV register. See Section 33.6.6 “Time-out value register”.
- Set the SCL high and low times to complete the bus rate setup. See Section 33.6.9 “Master control register”.

33.4.1.1 Master write to slave

Configure Flexcomm Interface 1 as I2C interface, see Chapter 25 “LPC5500 Flexcomm Interface serial communication”.

Configure the I²C as a master: set the MSTEN bit to 1 in the CFG register. See Table 622.

Write data to the slave:

1. Write the slave address with the $\overline{RW}$ bit set to 0 to the master data register MSTDAT. See Table 634
2. Start the transmission by setting the MSTSTART bit to 1 in the master control register. See Table 631. The following happens:
   - The pending status is cleared and the I²C-bus is busy.
   - The I²C master sends the start bit and address with the $\overline{RW}$ bit to the slave.
3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register
4. .Write 8 bits of data to the MSTDAT register.
5. Continue with the transmission of data by setting the MSTCONT bit to 1 in the master control register. See Table 631. The following happens:
   - The pending status is cleared and the I²C-bus is busy.
   - The I²C master sends the data bits to the slave address.
6. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
7. Stop the transmission by setting the MSTSTOP bit to 1 in the master control register. See Table 631.

Table 617. Code example

Master write to slave

```
//Master write 1 byte to slave. Address 0x23, Data 0xdd. Polling mode.
I2C->CFG = I2C_CFG_MSTEN;
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
I2C->MSTDAT = (0x23 << 1) | 0; // address and 0 for RWn bit
I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_TX) abort();
I2C->MSTDAT = 0xdd; // send data
I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_TX) abort();
I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
```

33.4.1.2 Master read from slave

Configure Flexcomm Interface 1 as I²C interface, see Chapter 25 “LPC5500 Flexcomm Interface serial communication”.

Configure the I²C as a master: set the MSTEN bit to 1 in the CFG register. See Table 623.

Read data from the slave:

Write the slave address with the $\overline{RW}$ bit set to 1 to the master data register MSTDAT. See Table 634

Start the transmission by setting the MSTSTART bit to 1 in the master control register. See Table 631. The following happens:

The pending status is cleared and the I²C-bus is busy.

The I²C master sends the start bit and address with the RW bit to the slave.

The slave sends eight bit of data.

Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.

Read eight bits of data from the MSTDAT register.

Stop the transmission by setting the MSTSTOP bit to 1 in the master control register. See Table 631.

Table 618. Code example

Master read from slave

```
// Master read 1 byte from slave. Address 0x23. Polling mode. No error checking.
uint8_t data;
I2C->CFG = I2C_CFG_MSTEN;
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
I2C->MSTDAT = (0x23 << 1) | 1; // address and 1 for RWn bit
I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort();
data = I2C->MSTDAT; // read data
if(data != 0xdd) abort();
I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
```

33.4.2 I²C receive/transmit in slave mode

In this example, Flexcomm Interface 1 is configured as an I²C slave. The slave receives 8 bits from the master and then sends 8 bits to the master. The SCL and SDA functions must be enabled on pins PIO0_22 and PIO0_23 through IOCON. See Section 15.5.2 “Type I IOCON registers”.

The pins should be configured as required for the I²C-bus mode that will be used (SM, FM, FM+, HS) via the IOCON block. See Section 15.5.2 “Type I IOCON registers”.

The transmission of the address and data bits is controlled by the state of the SLVPENDING status bit. Whenever the status is slave pending, the slave can acknowledge (*ack*) or send or receive an address and data. The received data or the data

to be sent to the master are available in the SLVDAT register. After sending and receiving data, continue to the next step of the transmission protocol by writing to the SLVCTL register

33.4.2.1 Slave read from master

Configure Flexcomm Interface 1 as I²C interface, see Chapter 32 "LPC55S6x/LPC55S2x/LPC552x Flexcomm Interface Serial Communication" Configure the I²C as a slave with address x:

Set the SLVEN bit to 1 in the CFG register. See Table 623.

- Write the slave address x to the address 0 match register. See Table 636.

Read data from the master:

1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
2. Acknowledge (*ack*) the address by setting SLVCONTINUE = 1 in the slave control register. See Table 635.
3. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
4. Read 8 bits of data from the SLVDAT register. See Table 635.

Acknowledge (*ack*) the data by setting SLVCONTINUE = 1 in the slave control register. See Table 634.

Table 619. Code example

Slave read from master

```
//Slave read 1 byte from master. Address 0x23. Polling mode.
uint8_t data;
I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register
I2C->CFG = I2C_CFG_SLVEN;
I2C->CFG;
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort();
data = I2C->SLVDAT; // read data
if(data != 0xdd) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack data
```

33.4.2.2 Slave write to master

Configure Flexcomm Interface 1 as I²C interface, Chapter 32 "LPC55S6x/LPC55S2x/LPC552x Flexcomm Interface Serial Communication" Configure the I²C as a slave with address x:

- Set the SLVEN bit to 1 in the CFG register. See Table 622.
- Write the slave address x to the address 0 match register. See Table 636.

Write data to the master:

1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.

2. ACK the address by setting SLVCONTINUE = 1 in the slave control register. See Table 634.
3. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
4. Write 8 bits of data to SLVDAT register. See Table 635.

Continue the transaction by setting SLVCONTINUE = 1 in the slave control register. See Table 634.

Table 620. Code example

Slave write to master

```
//Slave write 1 byte to master. Address 0x23, Data 0xdd. Polling mode.
I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register
I2C->CFG = I2C_CFG_SLVEN;
I2C->CFG;
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_TX) abort();
I2C->SLVDAT = 0xdd; // write data
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // continue transaction
```

33.4.3 Configure the I²C for wake-up

In sleep mode, any activity on the I²C-bus that triggers an I²C interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the Flexcomm Interface clock remains active in sleep mode, the I²C can wake up the part independently of whether the I²C interface is configured in master or slave mode.

In deep-sleep mode, the I²C clock is turned off as are all peripheral clocks. However, if the I²C is configured in slave mode and an external master on the I²C-bus provides the clock signal, the I²C interface can create an interrupt asynchronously. This interrupt, if enabled via the POWER_EnterDeepSleep() low power API and in the I²C interface INTENCLR register, can then wake up the core.

33.4.3.1 Wake-up from sleep mode

- Enable the I²C interrupt in the NVIC.
- Enable the I²C wake-up event in the INTENSET register. Wake-up on any enabled interrupts is supported (see the INTENSET register). Examples are the following events:
  - Master pending
  - Change to idle state
  - Start/stop error
  - Slave pending
  - Address match (in slave mode)
  - Data available/ready

33.4.3.2 Wake-up from deep-sleep mode

- Enable the I²C interrupt in the NVIC.
- Enable the I²C interrupt using low power API to create the interrupt signal asynchronously while the core and the peripheral are not clocked.
- Configure the I²C in slave mode.
- Enable the I²C the interrupt in the INTENCLR register which configures the interrupt as wake-up event. Examples are the following events:
  - Slave deselect
  - Slave pending (wait for read, write, or ACK)
  - Address match
  - Data available/ready for the monitor function.

33.5 General description

The architecture of the I²C-bus interface is shown in Figure 116.

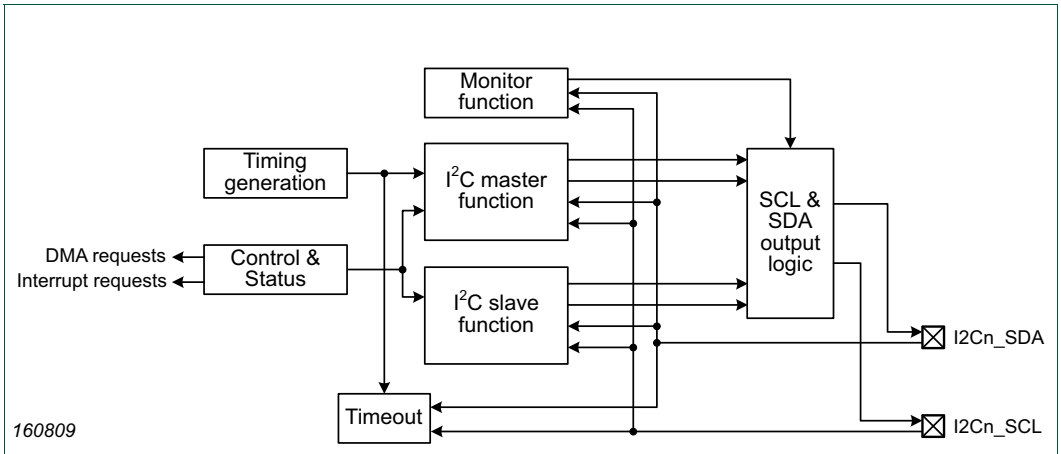


Fig 116. I²C block diagram

33.6 Register description

Address offsets are within the address space of the related Flexcomm Interface. The reset value reflects the data stored in used bits only. It does not include reserved bits content.

33.6.1 FLEXCOMM memory map

FLEXCOMM0 base address: 4008 6000h

FLEXCOMM1 base address: 4008 7000h

FLEXCOMM2 base address: 4008 8000h

FLEXCOMM3 base address: 4008 9000h

FLEXCOMM4 base address: 4008 A000h

FLEXCOMM5 base address: 4009 6000h

FLEXCOMM6 base address: 4009 7000h

FLEXCOMM7 base address: 4009 8000h

Table 621. Register overview: I2C register

Name	Access	Offset	Description	Reset value	Section
Shared I2C registers:					
CFG	R/W	0x800	Configuration for shared functions.	0	33.6.2
STAT	R/W	0x804	Status register for master, slave, and monitor functions.	0x0801	33.6.3
INTENSET	R/W	0x808	Interrupt enable set and read.	0	33.6.4
INTENCLR	WO	0x80C	Interrupt enable clear.	NA	33.6.5
TIMEOUT	R/W	0x810	Time-out value.	0xFFFF	33.6.6
CLKDIV	R/W	0x814	Clock pre-divider for the entire I2C interface. This determines what time increments are used for the MSTTIME register, and controls some timing of the slave function.	0	33.6.7
INTSTAT	RO	0x818	Interrupt status register for master, slave, and monitor functions.	0	33.6.8
Master function registers:					
MSTCTL	R/W	0x820	Master control.	0	33.6.9
MSTTIME	R/W	0x824	Master timing configuration.	0x0	33.6.10
MSTDAT	R/W	0x828	Combined master receiver and transmitter data.	NA	33.6.11
Slave function registers:					
SLVCTL	R/W	0x840	Slave control.	0	33.6.12
SLVDAT	R/W	0x844	Combined slave receiver and transmitter data.	NA	33.6.13
SLVADR0	R/W	0x848	Slave address 0.	0x01	33.6.14
SLVADR1	R/W	0x84C	Slave address 1.	0x01	33.6.15
SLVADR2	R/W	0x850	Slave address 2.	0x01	33.6.15
SLVADR3	R/W	0x854	Slave address 3.	0x01	33.6.15
SLVQUAL0	R/W	0x858	Slave qualification for address 0.	0	33.6.16

Table 621. Register overview: I2C register *…continued*

Name	Access	Offset	Description	Reset value	Section
Monitor function registers:					
MONRXDAT	RO	0x880	Monitor receiver data.	0	33.6.17
ID register:					
ID	RO	0xFFC	I2C module Identification. This value appears in the shared Flexcomm Interface peripheral ID register when I2C is selected.	0xE030 1300	33.6.18

33.6.2 I2C configuration register

The CFG register contains mode settings that apply to master, slave, and monitor functions.

Table 622. I2C configuration register (CFG, offset = 0x800)

Bit	Symbol	Value	Description	Reset value
0	MSTEN		Master enable. When disabled, configurations settings for the master function are not changed, but the master function is internally reset.	0
		0	Disabled. The I2C master function is disabled.	
		1	Enabled. The I2C master function is enabled.	
1	SLVEN		Slave enable. When disabled, configurations settings for the slave function are not changed, but the slave function is internally reset.	0
		0	Disabled. The I2C slave function is disabled.	
		1	Enabled. The I2C slave function is enabled.	
2	MONEN		Monitor enable. When disabled, configurations settings for the monitor function are not changed, but the monitor function is internally reset.	0
		0	Disabled. The I2C monitor function is disabled.	
		1	Enabled. The I2C monitor function is enabled.	
3	TIMEOUTEN		I2C bus time-out enable. When disabled, the time-out function is internally reset.	0
		0	Disabled. Time-out function is disabled.	
		1	Enabled. Time-out function is enabled. Both types of time-out flags will be generated and will cause interrupts if they are enabled. Typically, only one time-out will be used in a system.	
4	MONCLKSTR		Monitor function clock stretching.	0
		0	Disabled. The monitor function will not perform clock stretching. Software or DMA may not always be able to read data provided by the monitor function before it is overwritten. This mode may be used when non-invasive monitoring is critical.	
		1	Enabled. The monitor function will perform clock stretching in order to ensure that software or DMA can read all incoming data supplied by the monitor function.	

Table 622. I2C configuration register (CFG, offset = 0x800) *…continued*

Bit	Symbol	Value	Description	Reset value
5	HSCAPABLE		High-speed mode capable enable. Since high-speed mode alters the way I2C pins drive and filter, as well as the timing for certain I2C signalling, enabling high-speed mode applies to all functions: master, slave, and monitor.	0
		0	Standard or Fast-modes. The I2C interface will support Standard-mode, Fast-mode, and Fast-mode Plus, to the extent that the pin electronics support these modes. Any changes that need to be made to the pin controls, such as changing the drive strength or filtering, must be made by software via the IOCON register associated with each I2C pin,	
		1	High-speed. In addition to Standard-mode, Fast-mode, and Fast-mode Plus, the I2C interface will support high-speed mode to the extent that the pin electronics support these modes. See Section 33.7.2.2 “Bus rate support” for more information.	
31:6	-	-	Reserved. Read value is undefined, only zero should be written.	-

33.6.3 I2C status register

The STAT register provides status flags and state information about all of the functions of the I2C interface. Access to bits in this register varies. RO = read-only, W1C = write 1 to clear.

Details of the master and slave states described in the MSTSTATE and SLVSTATE bits in this register are listed in Table 625 and Table 626.

Table 623. I2C status register (STAT, offset = 0x804)

Bit	Symbol	Value	Description	Reset value	AccesS
0	MSTPENDING		Master Pending. Indicates that the master is waiting to continue communication on the I2C-bus (pending) or is idle. When the master is pending, the MSTSTATE bits indicate what type of software service if any the master expects. This flag will cause an interrupt when set if, enabled via the INTENSET register. The MSTPENDING flag is not set when the DMA is handling an event (if the MSTDMA bit in the MSTCTL register is set). If the master is in the idle state, and no communication is needed, mask this interrupt.	1	RO
		0	In progress. Communication is in progress and the master function is busy and cannot currently accept a command.		
		1	Pending. The master function needs software service or is in the idle state. If the master is not in the idle state, it is waiting to receive or transmit data or the NACK bit.		

Table 623. I2C status register (STAT, offset = 0x804) *…continued*

Bit	Symbol	Value	Description	Reset value	AccesS
3:1	MSTSTATE		Master State code. The master state code reflects the master state when the MSTPENDING bit is set, that is the master is pending or in the idle state. Each value of this field indicates a specific required service for the master function. All other values are reserved. See Table 625 for details of state values and appropriate responses.	0	RO
		0x0	Idle. The master function is available to be used for a new transaction.		
		0x1	Receive ready. Received data available (master receiver mode). Address plus read was previously sent and acknowledged by slave.		
		0x2	Transmit ready. Data can be transmitted (master transmitter mode). Address plus write was previously sent and acknowledged by slave.		
		0x3	NACK address. Slave NACKed address.		
		0x4	NACK Data. Slave NACKed transmitted data.		
4	MSTARBLOSS		Master Arbitration Loss flag. This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically a 1 is written to MSTCONTINUE.	0	W1C
		0	No Arbitration Loss has occurred.		
		1	Arbitration loss. The master function has experienced an arbitration loss. At this point, the master function has already stopped driving the bus and gone to an idle state. Software can respond by doing nothing, or by sending a start in order to attempt to gain control of the bus when it next becomes idle.		
5	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
6	MSTSTSTPERR		Master start/stop error flag. This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically a 1 is written to MSTCONTINUE.	0	W1C
		0	No start/stop error has occurred.		
		1	The master function has experienced a start/stop error. A start or stop was detected at a time when it is not allowed by the I2C specification. The master interface has stopped driving the bus and gone to an idle state, no action is required. A request for a start could be made, or software could attempt to insure that the bus has not stalled.		
7	-	-	Reserved. Read value is undefined, only zero should be written.	-	-

Table 623. I2C status register (STAT, offset = 0x804) *…continued*

Bit	Symbol	Value	Description	Reset value	AccesS
8	SLVPENDING		Slave Pending. Indicates that the slave function is waiting to continue communication on the I2C-bus and needs software service. This flag will cause an interrupt when set if enabled via INTENSET. The SLVPENDING flag is not set when the DMA is handling an event (if the SLVDMA bit in the SLVCTL register is set). The SLVPENDING flag is read-only and is automatically cleared when a 1 is written to the SLVCONTINUE bit in the SLVCTL register. The point in time when SlvPending is set depends on whether the I2C interface is in HSCAPABLE mode. See Section 33.7.2.2 “Bus rate support”. When the I2C interface is configured to be HSCAPABLE, HS master codes are detected automatically. Due to the requirements of the HS I2C specification, slave addresses must also be detected automatically, since the address must be acknowledged before the clock can be stretched.	0	RO
		0	In progress. The slave function does not currently need service.		
		1	Pending. The slave function needs service. Information on what is needed can be found in the adjacent SLVSTATE field.		
10:9	SLVSTATE		Slave State code. Each value of this field indicates a specific required service for the slave function. All other values are reserved. See Table 626 for state values and actions. **Remark:** note that the occurrence of some states and how they are handled are affected by DMA mode and Automatic Operation modes.	0	RO
		0x0	Slave address. Address plus R/W received. At least one of the four slave addresses has been matched by hardware.		
		0x1	Slave receive. Received data is available (slave receiver mode).		
		0x2	Slave transmit. Data can be transmitted (slave transmitter mode).		
11	SLVNOTSTR		Slave Not Stretching. Indicates when the slave function is stretching the I2C clock. This is needed in order to gracefully invoke deep-sleep mode during slave operation. This read-only flag reflects the slave function status in real time.	1	RO
		0	Stretching. The slave function is currently stretching the I2C bus clock. Deep-sleep mode cannot be entered at this time.		
		1	Not stretching. The slave function is not currently stretching the I2C bus clock. Deep-sleep mode could be entered at this time.		

Table 623. I2C status register (STAT, offset = 0x804) *…continued*

Bit	Symbol	Value	Description	Reset value	AccesS
13:12	SLVIDX		Slave address match Index. This field is valid when the I2C slave function has been selected by receiving an address that matches one of the slave addresses defined by any enabled slave address registers, and provides an identification of the address that was matched. It is possible that more than one address could be matched, but only one match can be reported here.	0	RO
		0x0	Address 0. Slave address 0 was matched.		
		0x1	Address 1. Slave address 1 was matched.		
		0x2	Address 2. Slave address 2 was matched.		
		0x3	Address 3. Slave address 3 was matched.		
14	SLVSEL		Slave selected flag. SLVSEL is set after an address match when software tells the slave function to acknowledge the address, or when the address has been automatically acknowledged. It is cleared when another address cycle presents an address that does not match an enabled address on the slave function, when slave software decides to NACK a matched address, when there is a stop detected on the bus, when the master NACKs slave data, and in some combinations of Automatic Operation. SLVSEL is not cleared if software NACKs data.	0	RO
		0	Not selected. The slave function is not currently selected.		
		1	Selected. The slave function is currently selected.		
15	SLVDESEL		Slave Deselected flag. This flag will cause an interrupt when set if enabled via INTENSET. This flag can be cleared by writing a 1 to this bit.	0	W1C
		0	Not deselected. The slave function has not become deselected. This does not mean that it is currently selected. That information can be found in the SLVSEL flag.		
		1	Deselected. The slave function has become deselected. This is specifically caused by the SLVSEL flag changing from 1 to 0. See the description of SLVSEL for details on when that event occurs.		
16	MONRDY		Monitor Ready. This flag is cleared when the MONRXDAT register is read.	0	RO
		0	No data. The monitor function does not currently have data available.		
		1	Data waiting. The monitor function has data waiting to be read.		
17	MONOV		Monitor Overflow flag.	0	W1C
		0	No overrun. Monitor data has not overrun.		
		1	Overrun. A monitor data overrun has occurred. This can only happen when monitor clock stretching not enabled via the MONCLKSTR bit in the CFG register. Writing 1 to this bit clears the flag.		

Table 623. I2C status register (STAT, offset = 0x804) *…continued*

Bit	Symbol	Value	Description	Reset value	AccesS
18	MONACTIVE		Monitor active flag. Indicates when the monitor function considers the I2C bus to be active. Active is defined here as when some master is on the bus: a bus start has occurred more recently than a bus stop.	0	RO
		0	Inactive. The monitor function considers the I2C bus to be inactive.		
		1	Active. The monitor function considers the I2C bus to be active.		
19	MONIDLE		Monitor idle flag. This flag is set when the monitor function sees the I2C bus change from active to inactive. This can be used by software to decide when to process data accumulated by the monitor function. This flag will cause an interrupt when set if enabled via the INTENSET register. The flag can be cleared by writing a 1 to this bit.	0	W1C
		0	Not idle. The I2C bus is not idle, or this flag has been cleared by software.		
		1	Idle. The I2C bus has gone idle at least once since the last time this flag was cleared by software.		
23:20	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
24	EVENTTIMEOUT		Event Time-out interrupt flag. Indicates when the time between events has been longer than the time specified by the TIMEOUT register. Events include start, stop, and clock edges. The flag is cleared by writing a 1 to this bit. No time-out is created when the I2C-bus is idle.	0	W1C
		0	No time-out. I2C bus events have not caused a time-out.		
		1	Event time-out. The time between I2C bus events has been longer than the time specified by the TIMEOUT register.		
25	SCLTIMEOUT		SCL Time-out interrupt flag. Indicates when SCL has remained low longer than the time specific by the TIMEOUT register. The flag is cleared by writing a 1 to this bit.	0	W1C
		0	No time-out. SCL low time has not caused a time-out.		
31:26	-	-	Reserved. Read value is undefined, only zero should be written.	-	

Table 624. Master function state codes (MSTSTATE)

MST STATE	Description	Actions	DMA allowed
0x0	**Idle.** The master function is available to be used for a new transaction.	Send a start or disable MSTPENDING interrupt if the master function is not needed currently.	No
0x1	**Received data is available (master receiver mode).** Address plus read was previously sent and acknowledged by slave.	Read data and either continue, send a stop, or send a repeated start.	Yes
0x2	**Data can be transmitted (master transmitter mode).** Address plus write was previously sent and acknowledged by slave.	Send data and continue, or send a stop or repeated start.	Yes
0x3	**Slave NACKed address.**	Send a stop or repeated start.	No
0x4	**Slave NACKed transmitted data.**	Send a stop or repeated start.	No

Table 625. Slave function state codes (SLVSTATE)

SLVSTATE		Description	Actions	DMA allowed
0	SLVST_ ADDR	**Address plus R/W received.** At least one of the 4 slave addresses has been matched by hardware.	Software can further check the address if needed, for instance if a subset of addresses qualified by SLVQUAL0 is to be used. Software can ACK or NACK the address by writing 1 to either SLVCONTINUE or SLVNACK. Also see Section 33.7.4 “Ten-bit addressing” regarding 10-bit addressing.	No
1	SLVST_RX	**Received data is available (slave receiver mode).**	Read data, reply with an ACK or a NACK.	Yes
2	SLVST_TX	**Data can be transmitted (slave transmitter mode).**	Send data. Note that when the master NACKs dat transmitted by the slave, the slave becomes de-selected.	Yes

33.6.4 Interrupt enable set and read register

The INTENSET register controls which I²C status flags generate interrupts. Writing a 1 to a bit position in this register enables an interrupt in the corresponding position in the STAT register Table 624, if an interrupt is supported there. Reading INTENSET indicates which interrupts are currently enabled.

Table 626. Interrupt enable set and read register (INTENSET, offset = 0x808)

Bit	Symbol	Value	Description	Reset value
0	MSTPENDINGEN		Master pending interrupt enable.	0
		0	Disabled. The MstPending interrupt is disabled.	
		1	Enabled. The MstPending interrupt is enabled.	
3:1	-	-	Reserved. Read value is undefined, only zero should be written.	-
4	MSTARBLOSSEN		Master arbitration loss interrupt enable.	0
		0	Disabled. The MstArbLoss interrupt is disabled.	
		1	Enabled. The MstArbLoss interrupt is enabled.	
5	-	-	Reserved. Read value is undefined, only zero should be written.	-

Table 626. Interrupt enable set and read register (INTENSET, offset = 0x808) *…continued*

Bit	Symbol	Value	Description	Reset value
6	MSTSTSTPERREN		Master start/stop error interrupt enable.	0
		0	Disabled. The MstStStpErr interrupt is disabled.	
		1	Enabled. The MstStStpErr interrupt is enabled.	
7	-		Reserved. Read value is undefined, only zero should be written.	-
8	SLVPENDINGEN		Slave Pending interrupt enable.	0
		0	Disabled. The SlvPending interrupt is disabled.	
		1	Enabled. The SlvPending interrupt is enabled.	
10:9	-	-	Reserved. Read value is undefined, only zero should be written.	-
11	SLVNOTSTREN		Slave Not Stretching interrupt enable.	0
		0	Disabled. The SlvNotStr interrupt is disabled.	
		1	Enabled. The SlvNotStr interrupt is enabled.	
14:12	-	-	Reserved. Read value is undefined, only zero should be written.	-
15	SLVDESELEN		Slave deselect interrupt enable.	0
		0	Disabled. The SlvDeSel interrupt is disabled.	
		1	Enabled. The SlvDeSel interrupt is enabled.	
16	MONRDYEN		Monitor data Ready interrupt enable.	0
		0	Disabled. The MonRdy interrupt is disabled.	
		1	Enabled. The MonRdy interrupt is enabled.	
17	MONOVEN		Monitor Overrun interrupt enable.	0
		0	Disabled. The MonOv interrupt is disabled.	
		1	Enabled. The MonOv interrupt is enabled.	
18	-	-	Reserved. Read value is undefined, only zero should be written.	-
19	MONIDLEEN		Monitor Idle interrupt enable.	0
23:20	-	-	Reserved. Read value is undefined, only zero should be written.	-
24	EVENTTIMEOUTEN		Event time-out interrupt enable.	0
		0	Disabled. The Event time-out interrupt is disabled.	
		1	Enabled. The Event time-out interrupt is enabled.	
25	SCLTIMEOUTEN		SCL time-out interrupt enable.	0
		0	Disabled. The SCL time-out interrupt is disabled.	
		1	Enabled. The SCL time-out interrupt is enabled.	
31:26	-	-	Reserved. Read value is undefined, only zero should be written.	-

33.6.5 Interrupt enable clear register

Writing a 1 to a bit position in INTENCLR clears the corresponding position in the INTENSET register, disabling that interrupt. INTENCLR is a write-only register.

Bits that do not correspond to defined bits in INTENSET are reserved and only Zeros should be written to them.

Table 627. Interrupt enable clear register (INTENCLR, offset = 0x80C)

Bit	Symbol	Description	Reset value
0	MSTPENDINGCLR	Master pending interrupt clear. Writing 1 to this bit clears the corresponding bit in the INTENSET register if implemented.	0
3:1	-	Reserved. Read value is undefined, only zero should be written.	-
4	MSTARBLOSSCLR	Master arbitration loss interrupt clear.	0
5	-	Reserved. Read value is undefined, only zero should be written.	-
6	MSTSTSTPERRCLR	Master start/stop error interrupt clear.	0
7	-	Reserved. Read value is undefined, only zero should be written.	-
8	SLVPENDINGCLR	Slave pending interrupt clear.	0
10:9	-	Reserved. Read value is undefined, only zero should be written.	-
11	SLVNOTSTRCLR	Slave Not Stretching interrupt clear.	0
14:12	-	Reserved. Read value is undefined, only zero should be written.	-
15	SLVDESELCLR	Slave deselect interrupt clear.	0
16	MONRDYCLR	Monitor data ready interrupt clear.	0
17	MONOVCLR	Monitor overrun interrupt clear.	0
18	-	Reserved. Read value is undefined, only zero should be written.	-
19	MONIDLECLR	Monitor Idle interrupt clear.	0
23:20	-	Reserved. Read value is undefined, only zero should be written.	-
24	EVENTTIMEOUTCLR	Event time-out interrupt clear.	0
25	SCLTIMEOUTCLR	SCL time-out interrupt clear.	0
31:26	-	Reserved. Read value is undefined, only zero should be written.	-

33.6.6 Time-out value register

The TIMEOUT register allows setting an upper limit to certain I2C bus times, informing by status flag and/or interrupt when those times are exceeded.

Two time-outs are generated, and software can elect to use either of them.

1. EVENTTIMEOUT checks the time between bus events while the bus is not idle: start, SCL rising, SCL falling, and stop. The EVENTTIMEOUT status flag in the STAT register is set if the time between any two events becomes longer than the time configured in the TIMEOUT register. The EVENTTIMEOUT status flag can cause an interrupt if enabled to do so by the EVENTTIMEOUTEN bit in the INTENSET register.
2. SCLTIMEOUT checks only the time that the SCL signal remains low while the bus is not idle. The SCLTIMEOUT status flag in the STAT register is set if SCL remains low longer than the time configured in the TIMEOUT register. The SCLTIMEOUT status flag can cause an interrupt if enabled to do so by the SCLTIMEOUTEN bit in the INTENSET register. The SCLTIMEOUT can be used with the SMBus.

Also see Section 33.7.3 “Time-out”.

Table 628. Time-out value register (TIMEOUT, offset 0x810)

Bit	Symbol	Description	Reset value
3:0	TOMIN	Time-out time value, bottom four bits. These are hard-wired to 0xF. This gives a minimum time-out of 16 I2C function clocks and also a time-out resolution of 16 I2C function clocks.	0xF
15:4	TO	Time-out time value. Specifies the time-out interval value in increments of 16 I2C function clocks, as defined by the CLKDIV register. To change this value while I2C is in operation, disable all time-outs, write a new value to TIMEOUT, then re-enable time-outs. 0x000 = A time-out will occur after 16 counts of the I2C function clock. 0x001 = A time-out will occur after 32 counts of the I2C function clock. ... 0xFFF = A time-out will occur after 65,536 counts of the I2C function clock.	0xFFF
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

33.6.7 Clock divider register

The CLKDIV register divides down the Flexcomm Interface clock (FCLK) to produce the I2C function clock that is used to time various aspects of the I2C interface. The I2C function clock is used for some internal operations in the I2C interface and to generate the timing required by the I2C bus specification, some of which are user configured in the MSTTIME register for master operation. Slave operation uses CLKDIV for some timing functions.

See Section 33.7.2.1 “Rate calculations”for details on bus rate setup.

Table 629. I2C clock divider register (CLKDIV, offset = 0x814)

Bit	Symbol	Description	Reset value
15:0	DIVVAL	This field controls how the Flexcomm Interface clock (FCLK) is used by the I2C functions that need an internal clock in order to operate. See Section 33.7.2.1 “Rate calculations”. 0x0000 = I2C clock divider provides FCLK divided by 1. 0x0001 = I2C clock divider provides FCLK divided by 2. 0x0002 = I2C clock divider provides FCLK divided by 3. ... 0xFFFF = I2C clock divider provides FCLK divided by 65,536.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

33.6.8 Interrupt status register

The INTSTAT register provides register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See Table 624 for detailed descriptions of the interrupt flags.

Table 630. I2C interrupt status register (INTSTAT, offset = 0x818)

Bit	Symbol	Description	Reset value
0	MSTPENDING	Master pending.	1
3:1	-	Reserved.	-
4	MSTARBLOSS	Master arbitration loss flag.	0
5	-	Reserved. Read value is undefined, only zero should be written.	-

Table 630. I2C interrupt status register (INTSTAT, offset = 0x818) *…continued*

Bit	Symbol	Description	Reset value
6	MSTSTSTPERR	Master start/stop error flag.	0
7	-	Reserved. Read value is undefined, only zero should be written.	-
8	SLVPENDING	Slave pending.	0
10:9	-	Reserved. Read value is undefined, only zero should be written.	-
11	SLVNOTSTR	Slave not stretching status.	1
14:12	-	Reserved. Read value is undefined, only zero should be written.	-
15	SLVDESEL	Slave deselected flag.	0
16	MONRDY	Monitor ready.	0
17	MONOV	Monitor overflow flag.	0
18	-	Reserved. Read value is undefined, only zero should be written.	-
19	MONIDLE	Monitor Idle flag.	0
23:20	-	Reserved. Read value is undefined, only zero should be written.	-
24	EVENTTIMEOUT	Event time-out interrupt flag.	0
25	SCLTIMEOUT	SCL time-out interrupt flag.	0
31:26	-	Reserved. Read value is undefined, only zero should be written.	-

33.6.9 Master control register

The MSTCTL register contains bits that control various functions of the I2C master interface. Only write to this register when the master is pending (MSTPENDING = 1 in the STAT register, see Table 624.

Software should always write a complete value to MSTCTL, and not OR new control bits into the register as is possible in other registers such as CFG. This is due to the fact that MSTSTART and MSTSTOP are not self-clearing flags. ORing in new data following a start or stop may cause undesirable side effects.

After an initial I2C start, MSTCTL should generally only be written when the MSTPENDING flag in the STAT register is set, after the last bus operation has completed. An exception is when DMA is being used and a transfer completes. In this case there is no MSTPENDING flag, and the MSTDMA control bit would be cleared by software potentially at the same time as setting either the MSTSTOP or MSTSTART control bit.

Remark: When in the idle or slave NACKed states, see Table 625, set the MSTDMA bit either with or after the MSTCONTINUE bit. MSTDMA can be cleared at any time.

Table 631. Master control register (MSTCTL, offset = 0x820)

Bit	Symbol	Value	Description	Reset value
0	MSTCONTINUE		Master continue. This bit is write-only.	0
		0	No effect.	
		1	Continue. Informs the master function to continue to the next operation. This must done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation.	
1	MSTSTART		Master start control. This bit is write-only.	0
		0	No effect.	
		1	Start. A start will be generated on the I2C bus at the next allowed time.	

Table 631. Master control register (MSTCTL, offset = 0x820) *…continued*

Bit	Symbol	Value	Description	Reset value
2	MSTSTOP		Master stop control. This bit is write-only.	0
		0	No effect.	
		1	Stop. A stop will be generated on the I2C bus at the next allowed time, preceded by a NACK to the slave if the master is receiving data from the slave (master receiver mode).	
3	MSTDMA		Master DMA enable. Data operations of the I2C can be performed with DMA. Protocol type operations such as start, address, stop, and address match must always be done with software, typically via an interrupt. Address acknowledgement must also be done by software except when the I2C is configured to be HSCAPABLE (and address acknowledgement is handled entirely by hardware) or when Automatic Operation is enabled. When a DMA data transfer is complete, MSTDMA must be cleared prior to beginning the next operation, typically a start or stop.This bit is read/write.	0
		0	Disable. No DMA requests are generated for master operation.	
		1	Enable. A DMA request is generated for I2C master data operations. When this I2C master is generating acknowledge bits in master receiver mode, the acknowledge is generated automatically.	
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

33.6.10 Master time register

The MSTTIME register allows programming of certain times that may be controlled by the master function. These include the clock (SCL) high and low times, repeated start setup time, and transmitted data setup time.

The I2C clock pre-divider is described in Table 632.

Table 632. Master time register (MSTTIME, offset = 0x824)

Bit	Symbol	Value	Description	Reset value
2:0	MSTSCLLOW		Master SCL Low time. Specifies the minimum low time that will be asserted by this master on SCL. Other devices on the bus (masters or slaves) could lengthen this time. This corresponds to the parameter t_{LOW} in the I2C bus specification. I2C bus specification parameters t_{BUF} and $t_{SU;STA}$ have the same values and are also controlled by MSTSCLLOW.	7
		0x0	SCL low multiplier = 2. See Section 33.7.2.1 “Rate calculations”	
		0x1	SCL low multiplier = 3. See Section 33.7.2.1 “Rate calculations”	
		0x2	SCL low multiplier = 4. See Section 33.7.2.1 “Rate calculations”	
		0x3	SCL low multiplier = 5. See Section 33.7.2.1 “Rate calculations”	
		0x4	SCL low multiplier = 6. See Section 33.7.2.1 “Rate calculations”	
		0x5	SCL low multiplier = 7. See Section 33.7.2.1 “Rate calculations”	
		0x6	SCL low multiplier = 8. See Section 33.7.2.1 “Rate calculations”	
		0x7	SCL low multiplier = 9. See Section 33.7.2.1 “Rate calculations”	
3	-	-	Reserved.	-

Table 632. Master time register (MSTTIME, offset = 0x824) *…continued*

Bit	Symbol	Value	Description	Reset value
6:4	MSTSCLHIGH		Master SCL High time. Specifies the minimum high time that will be asserted by this master on SCL. Other masters in a multi-master system could shorten this time. This corresponds to the parameter t_{HIGH} in the I²C bus specification. I²C bus specification parameters $t_{SU;STO}$ and $t_{HD;STA}$ have the same values and are also controlled by MSTSCLHIGH.	7
		0x0	SCL high multiplier = 2. See Section 33.7.2.1 “Rate calculations”.	
		0x1	SCL high multiplier = 3. See Section 33.7.2.1 “Rate calculations”.	
		0x2	SCL high multiplier = 4. See Section 33.7.2.1 “Rate calculations”	
		0x3	SCL high multiplier = 5. See Section 33.7.2.1 “Rate calculations”	
		0x4	SCL high multiplier = 6. See Section 33.7.2.1 “Rate calculations”	
		0x5	SCL high multiplier = 7. See Section 33.7.2.1 “Rate calculations”	
		0x6	SCL high multiplier = 8. See Section 33.7.2.1 “Rate calculations”	
		0x7	SCL high multiplier = 9. See Section 33.7.2.1 “Rate calculations”	
31:7		-	Reserved. Read value is undefined, only zero should be written.	-

33.6.11 Master data register

The MSTDAT register provides the means to read the most recently received data for the master function, and to transmit data using the master function.

Table 633. Master data register (MSTDAT, offset = 0x828)

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

33.6.12 Slave control register

The SLVCTL register contains bits that control various functions of the I²C slave interface. Only write to this register when the slave is pending (SLVPENDING = 1 in the STAT register,Table 624.

Refer to Section 33.7.8 “Automatic operation”for details of the AUTOACK, AUTOMATCHREAD, and related settings.

Remark: When in the slave address state (slave state 0, see Table 626), set the SLVDMA bit either with or after the SLVCONTINUE bit. SLVDMA can be cleared at any time.

Table 634. Slave control register (SLVCTL, offset = 0x840)

Bit	Symbol	Value	Description	Reset Value
0	SLVCONTINUE		Slave continue.	0
		0	No effect.	
		1	Continue. Informs the Slave function to continue to the next operation, by clearing the SLVPENDING flag in the STAT register. This must be done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation. Automatic Operation has different requirements. SLVCONTINUE should not be set unless SLVPENDING = 1.	

Table 634. Slave control register (SLVCTL, offset = 0x840) *…continued*

Bit	Symbol	Value	Description	Reset Value
1	SLVNACK		Slave NACK.	0
		0	No effect.	
		1	NACK. Causes the slave function to NACK the master when the slave is receiving data from the master (slave receiver mode).	
2	-	-	Reserved. Read value is undefined, only zero should be written.	-
3	SLVDMA		Slave DMA enable.	0
		0	Disabled. No DMA requests are issued for slave mode operation.	
		1	Enabled. DMA requests are issued for I2C slave data transmission and reception.	
7:4	-	-	Reserved. Read value is undefined, only zero should be written.	-
8	AUTOACK		Automatic acknowledge.When this bit is set, it will cause an I2C header which matches SLVADR0 and the direction set by AUTOMATCHREAD to be ACKed immediately; this is used with DMA to allow processing of the data without intervention. If this bit is clear and a header matches SLVADR0, the behavior is controlled by AUTONACK in the SLVADR0 register: allowing NACK or interrupt.	0
		0	Normal, non-automatic operation. If AUTONACK = 0, an SlvPending interrupt is generated when a matching address is received. If AUTONACK = 1, received addresses are NACKed (ignored).	
		1	A header with matching SLVADR0 and matching direction as set by AUTOMATCHREAD will be ACKed immediately, allowing the master to move on to the data bytes. The ACK will clear this bit. If the address matches SLVADR0, but the direction does not match AUTOMATCHREAD, the behavior will depend on the AUTONACK bit in the SLVADR0 register: if AUTONACK is set, then it will be Nacked; else if AUTONACK is clear, then a SlvPending interrupt is generated.	
9	AUTOMATCHREAD		When AUTOACK is set, this bit controls whether it matches a read or write request on the next header with an address matching SLVADR0. Since DMA needs to be configured to match the transfer direction, the direction needs to be specified. This bit allows a direction to be chosen for the next operation.	0
		0	The expected next operation in Automatic mode is an I2C write.	
		1	The expected next operation in Automatic mode is an I2C read.	
31:10	-	-	Reserved. Read value is undefined, only zero should be written.	-

33.6.13 Slave data register

The SLVDAT register provides the means to read the most recently received data for the slave function and to transmit data using the slave function.

Table 635. Slave data register (SLVDAT, offset = 0x844)

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

33.6.14 Slave address 0 register

The SLVADR0 register allows enabling and defining one of the addresses that can be automatically recognized by the I2C slave hardware.

The I2C slave function has a total of 4 address comparators. The value in SLVADR0 can be qualified by the setting of the SLVQUAL0 register. The additional 3 address comparators do not include the address qualifier feature. For handling of the general call address, one of the 4 address registers can be programmed to respond to address 0.

Refer to Section 33.7.8 “Automatic operation”for details of AUTONACK and related settings.

Table 636. Slave address 0 register (SLVADR[0], offset = 0x848)

Bit	Symbol	Value	Description	Reset value
0	SADISABLE		Slave address 0 disable.	1
		0	Enabled. Slave address 0 is enabled.	
		1	Ignored slave address 0 is ignored.	
7:1	SLVADR		Slave address. Seven bit slave address that is compared to received addresses if enabled. The compare can be affected by the setting of the SLVQUAL0 register.	0
14:8	-	-	Reserved. Read value is undefined, only zero should be written.	-
15	AUTONACK		Automatic NACK operation. Used in conjunction with AUTOACK and AUTOMATCHREAD, allows software to ignore I2C traffic while handling previous I2C data or other operations.	0
		0	Normal operation, matching I2C addresses are not ignored.	
		1	Automatic-only mode. All incoming addresses are ignored (NACKed), unless AUTOACK is set, it matches SLVADR0, and AUTOMATCHREAD matches the direction.	
31:16	-	-	Reserved. Read value is undefined, only zero should be written.	-

33.6.15 Slave address 1, 2, and 3 registers

These slave address registers provide for three additional addresses that can be automatically recognized by the I2C slave hardware.

Table 637. Slave address registers (SLVADR[1:3], offset [0x84C:0x854])

Bit	Symbol	Value	Description	Reset value
0	SADISABLE		Slave address n disable.	1
		0	Enabled. Slave address n is enabled.	
		1	Ignored slave address n is ignored.	
7:1	SLVADR		Slave address. Seven bit slave address that is compared to received addresses if enabled.	0
31:8	-	-	Reserved. Read value is undefined, only zero should be written.	-

33.6.16 Slave address qualifier 0 register

The SLVQUAL0 register can alter how slave address 0 (specified by the SLVADR0 register) is interpreted.

Table 638. Slave address qualifier 0 register (SLVQUAL0, offset = 0x858)

Bit	Symbol	Value	Description	Reset Value
0	QUALMODE0		Qualify mode for slave address 0.	0
		0	Mask. The SLVQUAL0 field is used as a logical mask for matching address 0.	
		1	Extend. The SLVQUAL0 field is used to extend address 0 matching in a range of addresses.	
7:1	SLVQUAL0		Slave address Qualifier for address 0. A value of 0 causes the address in SLVADR0 to be used as-is, assuming that it is enabled. If QUALMODE0 = 0, any bit in this field which is set to 1 will cause an automatic match of the corresponding bit of the received address when it is compared to the SLVADR0 register. If QUALMODE0 = 1, an address range is matched for address 0. This range extends from the value defined by SLVADR0 to the address defined by SLVQUAL0 (address matches when SLVADR0[7:1] ≤ received address ≤ SLVQUAL0[7:1]).	0
31:8	-	-	Reserved. Read value is undefined, only zero should be written.	-

33.6.17 Monitor data register

The read-only MONRXDAT register provides information about events on the I2C bus, primarily to facilitate debugging of the I2C during application development. All data addresses and data passing on the bus and whether these were acknowledged, as well as start and stop events, are reported.

The monitor function must be enabled by the MONEN bit in the CFG register. Monitor mode can be configured to stretch the I2C clock if data is not read from the MONRXDAT register in time to prevent it, via the MONCLKSTR bit in the CFG register. This can help ensure that nothing is missed but can cause the monitor function to be somewhat intrusive (by potentially adding clock delays, depending on software or DMA response time). In order to improve the chance of collecting all monitor information if clock stretching is not enabled, monitor data is buffered such that it is available until the end of the next piece of information from the I2C bus.

Details of clock stretching are different in HS mode, see Section 33.7.2.2 “Bus rate support”.

Table 639. Monitor data register (MONRXDAT, offset = 0x880)

Bit	Symbol	Value	Description	Reset value
7:0	MONRXDAT		Monitor function receiver data. This reflects every data byte that passes on the I2C pins.	0
8	MONSTART		Monitor received start.	0
		0	No start detected. The monitor function has not detected a start event on the I2C bus.	
		1	Start detected. The monitor function has detected a start event on the I2C bus.	
9	MONRESTART		Monitor received repeated start.	0
		0	No repeated start detected. The monitor function has not detected a repeated start event on the I2C bus.	
		1	Repeated start detected. The monitor function has detected a repeated start event on the I2C bus.	

Table 639. Monitor data register (MONRXDAT, offset = 0x880)

Bit	Symbol	Value	Description	Reset value
10	MONNACK		Monitor received NACK.	0
		0	Acknowledged. The data currently being provided by the monitor function was acknowledged by at least one master or slave receiver.	
		1	Not acknowledged. The data currently being provided by the monitor function was not acknowledged by any receiver.	
31:11	-	-	Reserved. Read value is undefined, only zero should be written.	-

33.6.18 Module identification register

The ID register identifies the type and revision of the I2C module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 640. Module identification register (ID, offset = 0xFFC)

Bit	Symbol	Description	Reset Value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x00
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE030

33.7 Functional description

33.7.1 AHB bus access

The bus interface to the I2C registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the I2C function.

33.7.2 Bus rates and timing considerations

Due to the nature of the I2C bus, it is generally not possible to guarantee a specific clock rate on the SCL pin. On the I2C-bus, the clock can be stretched by any slave device, extended by software overhead time, etc.

In a multi-master system, the master that provides the shortest SCL high time will cause that time to appear on SCL as long as that master is participating in I2C traffic (i.e. when it is the only master on the bus, or during arbitration between masters).

In addition, I2C implementations generally base subsequent actions on what actually happens on the bus lines. For instance, a bus master allows SCL to go high. It then monitors the line to make sure it actually did go high (this would be required in a multi-master system). This results in a small delay before the next action on the bus, caused by the rise time of the open drain bus line.

Rate calculations give a base frequency that represents the fastest that the I2C bus could operate if nothing slows it down.

33.7.2.1 Rate calculations

Master timing

SCL high time (in Flexcomm Interface function clocks) =
I2C clock divider * SCL high multiplier, See Table 630 and Table 633

Nominal SCL rate =
Flexcomm Interface function clock rate / (SCL high time + SCL low time)

Remark: DIVVAL must be 1.

Remark: For 400 kHz clock rate, the clock frequency after the I2C divider (divval) must be ≤ 2 MHz. Table 642 shows the recommended settings for 400 kHz clock rate.

Table 641. Settings for 400 kHz clock rate

Input clock to I2C	DIVVAL for CLKDIV register	MSTSCLHIGH for MSTTIME register	MSTSCLLOW for MSTTIME register
30 MHz	14	0	1
24 MHz	14	0	0

Slave timing

Most aspects of slave operation are controlled by SCL received from the I2C bus master. However, if the slave function stretches SCL to allow for software response, it must provide sufficient data setup time to the master before releasing the stretched clock. This is accomplished by inserting one clock time of CLKDIV at that point.

If CLKDIV is already configured for master operation, that is sufficient. If only the slave function is used, CLKDIV should be configured such that one clock time is greater than the tSU;DAT value noted in the I2C bus specification for the I2C mode that is being used.

33.7.2.2 Bus rate support

The I2C interface can support 4 modes from the I2C bus specification:

- Standard-mode (SM, rate up to 100 kbits/s)
- Fast-mode (FM, rate up to 400 kbits/s)
- Fast-mode Plus (FM+, rate up to 1 Mbits/s)
- High-speed mode (HS, rate up to 3.4 Mbits/s)

The I2C interface supports Standard-mode, Fast-mode, and Fast-mode Plus with the same software sequence, which also supports SMBus. High-speed mode is intrinsically incompatible with SMBus due to conflicting requirements and limitations for clock stretching, and therefore requires a slightly different software sequence.

33.7.2.2.1 High-speed mode support

High-speed mode requires different pin filtering, somewhat different timing, and a different drive strength on SCL for the master function. The changes needed for the handling of the acknowledge bit mean that SMBus cannot be supported when the I2C is configured to be HS capable. This limitation is intrinsic to the SMBus and high-speed I2C specifications.

Because of the timing of changes to pin drive strength and filtering, the I2C interface is designed to directly control those pad characteristics when configured to be HS capable. The I2C also recognizes HS master codes and responds to programmed addresses when HS capable.

For software consistency, the changes required for handling of acknowledge and address recognition, and which affect when interrupts occur, are always in effect when the I2C is configured to be HS capable. This means that software does not need to know if a particular transfer is actually in HS mode or not.

33.7.2.2.2 Clock stretching

The I2C interface automatically stretches the clock when it does not have sufficient information on how to proceed, i.e. software has not supplied data and/or instructions to generate a start or stop. In principle, at least, I2C can allow the clock to be stretched by any bus participant at any time that SCL is low, in SM, FM, and FM+ modes.

In practice, the I2C interface described here may stretch SCL at the following times, in SM, FM, and FM+ modes:

- As a slave:
  - after an address is received that complies with at least one slave address (before the address is acknowledged)
  - as a slave receiver, after each data byte received (software then acknowledges the data)
  - as a slave transmitter, after each data byte is sent and the matching acknowledge is received from the master
- As a master:
  - after each address is sent and the acknowledge bit has been received
  - as a master receiver, after each after each data byte is received (software then acknowledges the data)
  - as a master transmitter, after each data byte is sent and the matching acknowledge bit has been received from the slave

In HS mode:

- As a slave (only slave functions in HS mode are supported on this device)
  - as a slave receiver, after each data byte is received and automatically acknowledged
  - as a slave transmitter, after each after each data byte is sent and the matching acknowledge is received from the master

In each case, the relevant pending flag (MSTPENDING or SLVPENDING) is set at the point where clock stretching occurs.

33.7.3 Time-out

A time-out feature on an I2C interface can be used to detect a *stuck* bus and potentially do something to alleviate the condition. Two different types of time-out are supported. Both types apply whenever the I2C interface and the time-out function are both enabled. master, slave, or monitor functions do not need to be enabled.

In the first type of time-out, reflected by the EVENTTIMEOUT flag in the STAT register, the time between bus events governs the time-out check. These events include start, stop, and all changes on the I2C clock (SCL). This time-out is asserted when the time between any of these events is longer than the time configured in the TIMEOUT register. This time-out could be useful in monitoring an I2C bus within a system as part of a method to keep the bus running of problems occur.

The second type of I2C time-out is reflected by the SCLTIMEOUT flag in the STAT register. This time-out is asserted when the SCL signal remains low longer than the time configured in the TIMEOUT register. This corresponds to SMBus time-out parameter $T_{TIMEOUT}$. In this situation, a slave could reset its own I2C interface in case it is the offending device. If all listening slaves (including masters that can be addressed as slaves) do this, then the bus will be released unless it is a current master causing the problem. Refer to the SMBus specification for more details.

Both types of time-out are generated only when the I2C bus is considered busy, i.e. when there has been a start condition more recently than a stop condition.

33.7.4 Ten-bit addressing

Ten-bit addressing is accomplished by the I2C master sending a second address byte to extend a particular range of standard 7-bit addresses. In the case of the master writing to the slave, the I2C frame simply continues with data after the two address bytes. For the master to read from a slave, it needs to reverse the data direction after the second address byte. It is done by sending a repeated start, followed by a repeat of the same standard 7-bit address, with a read bit. The slave must remember that it had been addressed by the previous write operation and stay selected for the subsequent read with the correct partial I2C address.

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

For the slave function, the first part of the address is automatically matched in the same fashion as 7-bit addressing. The slave address qualifier feature, , can be used to intercept all potential 10-bit addresses (first address byte values F0 through F6), or just one, see Section 33.6.15 “Slave address 1, 2, and 3 registers”. In the case of slave receiver mode, data is received in the normal fashion after software matches the first data byte to the remaining portion of the 10-bit address. The slave function should record the fact that it has been addressed, in case there is a follow-up read operation.

For slave transmitter mode, the slave function responds to the initial address in the same fashion as for slave receiver mode, and checks that it has previously been addressed with a full 10-bit address. If the address matched is address 0, and address qualification is enabled, software must check that the first part of the 10-bit address is a complete match to the previous address before acknowledging the address.

33.7.5 Clocking and power considerations

The master function of the I2C always requires a peripheral clock to be running in order to operate. The slave function can operate without any internal clocking when the slave is not currently addressed. This means that reduced power modes up to deep-sleep mode

can be entered, and the device will wake up when the I2C slave function recognizes an address. Monitor mode can similarly wake up the device from a reduced power mode when information becomes available.

33.7.6 Interrupt handling

The I2C provides a single interrupt output that handles all interrupts for master, slave, and monitor functions.

33.7.7 DMA

DMA with the I2C is done only for data transfer, DMA cannot handle control of the I2C. Once DMA is transferring data, I2C acknowledges are handled implicitly. No CPU intervention is required while DMA is transferring data.

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

DMA may be used in connection with automatic operation in order to minimize software overhead time for I2C handling.

A received NACK (from a slave in master mode, or from a master in slave mode) will cause DMA to stop and an interrupt to be generated. A repeated start sensed on the bus will similarly cause DMA to stop and an interrupt to be generated.

The monitor function may be used with DMA if a channel is available. See Section 22.5.1.1.1 “DMA with I2C monitor mode”for how DMA channels are used with the monitor function.

33.7.7.1 DMA as a master transmitter

A basic sequence for a master transmitter:

- Software sets up DMA to transmit a message.
- Software causes a slave address with write command to be sent and checks that the address was acknowledged.
- Software turns on DMA mode in the I2C.
- DMA transfers data and eventually completes the transfer.
- Software causes a stop (or repeated start) to be sent.

Software will be invoked to handle any exceptions to the standard transfer, such as the slave sending a NACK before the end of the transfer.

33.7.7.2 DMA as a master receiver

A basic sequence for a master receiver:

- Software sets up DMA to receive a message.
- Software causes a slave address with read command to be sent and checks that the address was acknowledged.

- Software starts DMA.
- DMA completes.
- Software causes a stop or repeated start to be sent.
- Software will be invoked to handle any exceptions to the standard transfer.

33.7.7.3 DMA as a slave transmitter

A basic sequence for a slave transmitter:

- Software acknowledges an I2C address.
- Software sets up DMA to transmit a message.
- Software starts DMA.
- DMA completes.

33.7.7.4 DMA as a slave receiver

A basic sequence for a slave receiver:

- Software receives an interrupt for a slave address received, and acknowledges the address.
- Software sets up DMA to receive a message, less the final data byte.
- Software starts DMA.
- DMA completes.
- Software sets SLVNACK prior to receiving the final data byte.
- Software receives the final data byte.

33.7.8 Automatic operation

Automatic operation modes provide a way to reduce software overhead for I2C slave functions with some limitations. They are intended to be used primarily in conjunction with slave DMA. Related control bits are SLVDMA, AUTOACK, and AUTOMATCHREAD in the SLCCTL register, and the AUTONACK bit in the SLVADR0 register. Table 27 shows how these controls may be used. These cases apply when an address matching SLVADR0, qualified by SLVQUAL0, is received.

Table 642. Automatic operation cases

Conditions:			Response:		
AUTONACK bit	**AUTOACK bit**	**Received R/W bit matches AUTOMATCHREAD**	**SLVPENDING interrupt generated?**	**ACK/NACK on I2C bus**	**Description**
0	0	x	Yes	None	Normal, non-automatic operation.
0	1	No	Yes	None	Automatic slave DMA: unexpected read/write case. Same as normal non-automatic operation.

Table 642. Automatic operation cases

Conditions:			Response:		
AUTONACK bit	**AUTOACK bit**	**Received R/W bit matches AUTOMATCHREAD**	**SLVPENDING interrupt generated?**	**ACK/NACK on I2C bus**	**Description**
x	1	Yes	Yes	ACK	Automatic slave DMA: expected read/write case. When the automatic ACK is sent, the SLVDMA bit is set and the AUTOACK bit is cleared.
1	0	x	No	NACK	Bus is ignored until software changes the setup.
1	1	No	No	NACK	Bus is ignored until software changes the setup.

34.1 How to read this chapter

USART functions are available on all LPC55S6x/LPC55S2x/LPC552x devices as a selectable function in each Flexcomm Interface peripheral. Up to 8 Flexcomm Interfaces are available.

34.2 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.
- The USART function supports separate transmit and receive FIFO with 16 entries each.
- 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.
- Break generation and detection.
- Receive data is 2 of 3 sample *voting*. Status flag set when one sample differs.
- Built-in baud rate generator.
- Autobaud mode for automatic baud rate detection.
- Special operating mode allows operation at up to 9600 baud using the 32 kHz RTC oscillator as the USART clock. This mode can be used while the device is in deep-sleep mode and can wake-up the device when a character is received.
- A fractional rate divider for USART.
- Interrupts available for FIFO receive level reached, FIFO transmit level reached, FIFO overflow or underflow, Transmitter Idle, change in receiver break detect, Framing error, Parity error, Delta CTS detect, and receiver sample noise detected (among others).
- USART transmit and receive functions can operated with the system DMA controller.
- Loopback mode for testing of data and flow control.

34.3 Basic configuration

Initial configuration of a USART peripheral is accomplished as follows:

- If needed, use the PRESETCTRL1 register, see Section 4.5.8 “Peripheral reset control 1” or Section 4.5.9 “Peripheral reset control 2” to reset the Flexcomm Interface that is about to have a specific peripheral function selected.
- Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface, see Section 32.7.1 “Peripheral Select and Flexcomm Interface ID register”.
- Configure the FIFOs for operation.
- Configure USART for receiving and transmitting data:
  - In the AHBCLKCTRL1 register, see Table 56, set the appropriate bit for the related Flexcomm Interface in order to enable the clock to the register interface.
  - Enable or disable the related Flexcomm Interface interrupt in the NVIC, see Table 8.
  - Configure the related Flexcomm Interface pin functions via IOCON, see Chapter 15 “LPC55S6x/LPC55S2x/LPC552x I/O Pin Configuration (IOCON)”.
  - Configure the Flexcomm Interface clock and USART baud rate. See Section 34.3.1 “Configure the Flexcomm Interface clock and USART baud rate”.

    Remark: The Flexcomm interface function clock frequency (at the output of the FRG associated with the Flexcomm) shall not be above **100 MHz**.

    Remark: The USART peripheral clock (the USART protocol bus clock) shall not be above 25 MHz (25 MBits/s maximum) in Synchronous Mode or above 10 MBits/s in Asynchronous Mode.
- Configure the USART to wake up the part from low power modes. See Section 34.3.2 “Configure the USART for wake-up”.
- Configure the USART to receive and transmit data in synchronous slave mode. See Section 34.3.2 “Configure the USART for wake-up”.

34.3.1 Configure the Flexcomm Interface clock and USART baud rate

Each Flexcomm Interface has a separate clock selection, which can include a shared fractional divider also see Section 34.7.2.3 “32 kHz mode”). The function clock and the fractional divider for the baud rate calculation are set up in the SYSCON block as follows:

1. If a fractional value is needed to obtain a particular baud rate, program the fractional rate divider (FRG, controlled by Syscon register FRGCTRL). 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.

   Flexcomm Interface clock = (FRG input clock) / (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.

   See Section 4.5.49 “Fractional rate divider for each Flexcomm Interface frequency” for more information on the FRG.

2. In asynchronous mode: configure the baud rate divider BRGVAL in the BRG register. The baud rate divider divides the Flexcomm Interface function clock (FCLK) to create the clock needed to produce the desired baud rate.

   Generally: baud rate = [FCLK / oversample rate] / BRG divide

   With specific register values: baud rate = [FCLK / (OSRVAL+1)] / (BRGVAL + 1)

   Generally: BRG divide = [FCLK / oversample rate] / baud rate

   With specific register values: BRGVAL = [[FCLK / (OSRVAL + 1)] / baud rate] - 1

   See Section 34.6.6 “USART baud rate generator register”.
3. In synchronous master mode: The serial clock is Un_SCLK = FCLK / (BRGVAL+1).

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 34.7.2.3 “32 kHz mode”.

To ensure that sync UART mode works at 25 MHz in Slave Receive Mode, uclk duty cycle must be 50%.

If FRG clock is used to derive the clock from the source clock, duty cycle is reduced to base on division factor (for example, 25% for divide-by-2). It reduces the time window available for data capture. Therefore, if 25 MHZ interface frequency is targeted, use a method other than FRG to derive interface clock. You can select PLL frequency and bypass FRG (MULT= 0x00 and DIV = 0xFF).

Note: If the USART BRG is set to 0, the FCLK input to the USART is sent directly to the SCLK pin in synchronous master mode. If the FRG is used to divide the source clock to produce the Flexcomm FCLK, that clock will not have a 50% duty cycle. Therefore, if the FRG is used, the BRG must also be set to divide the clock by some integer factor before it is used.

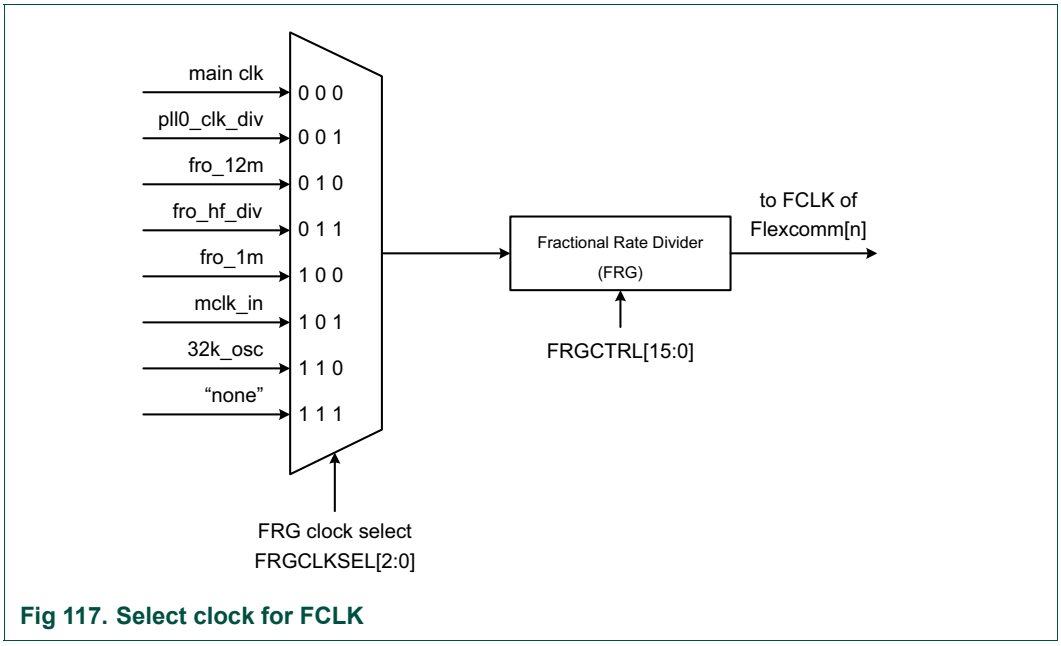


Fig 117. Select clock for FCLK

For details on the clock configuration see:

Section 34.7.2 “Clocking and baud rates”

34.3.2 Configure the USART for wake-up

A USART can wake up the system from sleep mode in asynchronous or synchronous mode on any enabled USART interrupt.

In deep-sleep mode, there are 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 on-chip clocks - that is in deep-sleep 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 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.

34.3.2.1 Wake-up from sleep mode

- Configure the USART in either asynchronous mode or synchronous mode. See Table 647.
- Enable the USART interrupt in the NVIC.
- Any enabled USART interrupt wakes up the part from sleep mode.

34.3.2.2 Wake-up from deep-sleep mode

- Configure the USART in synchronous slave mode. See Table 647. The SCLK function must be connected to a pin and also connect the pin to the master. Alternatively, the 32 kHz mode can be enabled and the USART operated in asynchronous mode with the 32 kHz RTC oscillator.
- Enable the USART interrupt using low power API
- Enable the USART interrupt in the NVIC.
- The USART wakes up the part from deep-sleep mode on all events that cause an interrupt and are enabled. Typical wake-up events are:
  - A start bit has been received.
  - Received data becomes available.
  - In synchronous mode, data is available in the FIFO to be transmitted, and a serial clock from the master is received.
  - A change in the state of the CTS pin if the CTS function is connected.

Remark: By enabling or disabling specific USART interrupts, you can customize when the wake-up occurs.

34.4 Pin description

The USART receive, transmit, and control signals are movable Flexcomm Interface functions and are assigned to external pins through via IOCON. See the IOCON description, see Chapter 15 “LPC55S6x/LPC55S2x/LPC552x I/O Pin Configuration (IOCON)” to assign the USART functions to pins on the device package. Recommended IOCON settings are shown in Table 644.

Table 643. USART pin description

Pin	Type	Name used in Pin Configuration chapter	Description
TXD	O	FCn_TXD_SCL_MISO_WS	Transmitter output for USART on Flexcomm Interface n. Serial transmit data.
RXD	I	FCn_RXD_SDA_MOSI_DATA	Receiver input for USART on Flexcomm Interface n. Serial receive data.
$\overline{\text{RTS}}$	O	FCn_RTS_SCL_SSEL1	Request To Send output for USART on Flexcomm Interface n. This signal supports inter-processor communication through the use of hardware flow control. This signal can also be configured to act as an output enable for an external RS-485 transceiver. RTS is active when the USART RTS signal is configured to appear on a device pin.
$\overline{\text{CTS}}$	I	FCn_CTS_SDA_SSEL0	Clear To Send input for USART on Flexcomm Interface n. 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).
SCLK	I/O	FCn_SCK	Serial clock input/output for USART on Flexcomm Interface n in synchronous mode. Clock input or output in synchronous mode. **Remark:** When the USART is configured as a master, such that SCK is an output, it must actually be connected to a pin in order for the USART to work properly.

Table 644. Suggested USART pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
31:16	Reserved	Reserved	Reserved
15	Reserved	Reserved	0b - Enabled. I2C 50 ns glitch filter enabled. 1b - Disabled. I2C 50 ns glitch filter disabled.
14	Reserved	Reserved	Select GPIO/I2C mode. Generally set to 1.
13	Reserved	Reserved	0b - Enabled. Pull resistor is connected. 1b - Disabled. IO is in open drain.
12	Reserved	Reserved	I2CFILTER: 0 for Fast or Standard mode I2C. 1 for Fast Mode Plus or high-speed slave

Table 644. Suggested USART pin settings *…continued*

IOCON bit(s)	Type D pin	Type A pin	Type I pin
11	Reserved	Reserved	SSEL: Generally set to 0
10	Reserved	ASW enable: Analog switch input control. Usable only if DIGIMODE = 0b0 Generally set to 0	Reserved
9	OD: Controls open-drain mode. 0b - Normal. Normal push-pull output. 1b - Open-drain. Simulated open-drain output (high drive disabled). Generally Set to 0 unless open drain is desired	Same as type D.	Same as type D.
8	DIGIMODE: 0b - Analog mode, digital input is disabled. 1b - Digital mode, digital input is enabled. Generally Set to 1.	Same as type D.	Same as type D.
7	INVERT: Input polarity. 0b - Disabled. Input function is not inverted. 1b -Enabled. Input is function inverted. Generally Set to 0.	Same as type D.	Same as type D.
6	SLEW, Driver slew rate. 0b - Standard mode, output slew rate control is enabled. More outputs can be switched simultaneously. 1b - Fast-mode, slew rate control is disabled. Generally Set to 0.	Same as type D.	Same as type D.

Table 644. Suggested USART pin settings *…continued*

IOCON bit(s)	Type D pin	Type A pin	Type I pin
5:4	MODE: Selects function mode (on-chip pull-up/pull-down resistor control). 00b - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01b - Pull-down. Pull-down resistor enabled. 10b - Pull-up. Pull-up resistor enabled. 11b - Repeater. Repeater mode. Generally Set to 0.	Same as type D.	Same as type D.
3:0	FUNC: Selects pin function. 0000b - Alternative connection 0. 01b - Pull-down. Pull-down resistor enabled. 0001b - Alternative connection 1. 0010b - Alternative connection 2. 0011b - Alternative connection 3. 0100b - Alternative connection 4. 0101b - Alternative connection 5. 0110b - Alternative connection 6. 0111b - Alternative connection 7.	Same as type D.	FUNC: The function will be “SCL” or “SDA”.
General comment	A good choice for USART input or output.	A reasonable choice for USART input or output.	Not recommended for USART functions that can be outputs in the chosen mode.

34.5 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 FIFO to await access by the CPU or DMA controller.

The USART transmitter block accepts data written by the CPU or DMA controller to the transmit FIFO. 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 an oversample clock (typically 16x) in the standard asynchronous operating mode. The BRG clock input source is the shared fractional rate generator that runs from the USART function clock. 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 provided via the STAT register. Many of the status flags are able to generate interrupts, as selected by software. The INTSTAT register provides a view of all interrupts that are both enabled and pending.

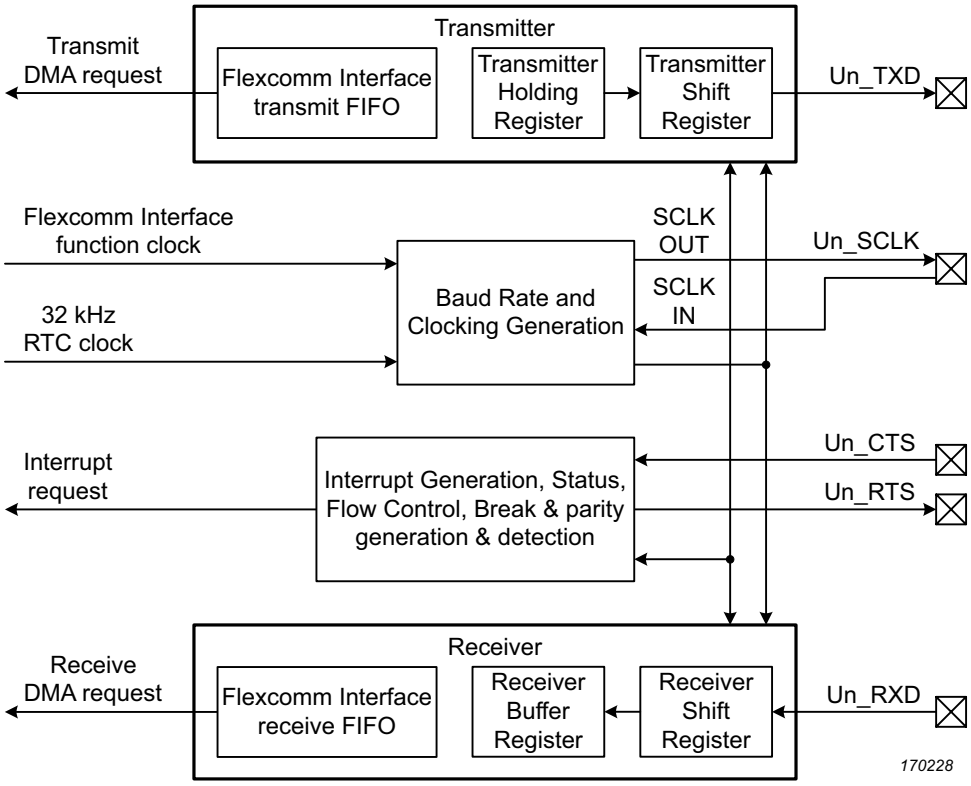


Fig 118. USART block diagram

34.6 Register description

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits. Address offsets are within the related Flexcomm Interface address space **after** the USART function is selected for that Flexcomm Interface, see Section 32.5.1 "Function Summary" for a summary of Flexcomm Interface addresses.

Table 645. USART base addresses

USART number	Base address
0	0x4008 6000h
1	0x4008 7000h
2	0x4008 8000h
3	0x4008 9000h
4	0x4008 A000h
5	0x4009 6000h
6	0x4009 7000h
7	0x4009 8000h

Table 646. USART register overview

Name	Access	Offset	Description	Reset value	Section
Registers for the USART function:					
CFG	R/W	0x000	USART Configuration register. Basic USART configuration settings that typically are not changed during operation.	0	34.6.1
CTL	R/W	0x004	USART Control register. USART control settings that are more likely to change during operation.	0	34.6.2
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.	0x1E	34.6.3
INTENSET	R/W	0x00C	Interrupt Enable read and Set register for USART (not FIFO) status. Contains individual interrupt enable bits 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	34.6.4
INTENCLR	WO	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.	-	34.6.5
BRG	R/W	0x020	Baud Rate Generator register. 16-bit integer baud rate divisor value.	0	34.6.6
INTSTAT	RO	0x024	Interrupt status register. Reflects interrupts that are currently enabled.	0x12	34.6.7
OSR	R/W	0x028	Oversample selection register for asynchronous communication.	0xF	34.6.8
ADDR	R/W	0x02C	Address register for automatic address matching.	0	34.6.9
Registers for FIFO control and data access:					
FIFOCFG	R/W	0xE00	FIFO configuration and enable register.	0	34.6.10
FIFOSTAT	R/W	0xE04	FIFO status register.	0x30	34.6.11
FIFOTRIG	R/W	0xE08	FIFO trigger settings for interrupt and DMA request.	0	34.6.12

Table 646. USART register overview *…continued*

Name	Access	Offset	Description	Reset value	Section
FIFOINTENSET	R/W1S	0xE10	FIFO interrupt enable set (enable) and read register.	0	34.6.13
FIFOINTENCLR	R/W1C	0xE14	FIFO interrupt enable clear (disable) and read register.	0	34.6.14
FIFOINTSTAT	RO	0xE18	FIFO interrupt status register.	0	34.6.15
FIFOWR	WO	0xE20	FIFO write data.	NA	34.6.16
FIFORD	RO	0xE30	FIFO read data.	NA	34.6.17
FIFORDNOPOP	RO	0xE40	FIFO data read with no FIFO pop.	NA	34.6.18
FIFOSIZE	R	0xE48	FIFO size.	0x10	34.6.19
ID register:					
ID	RO	0xFFC	USART module Identification. This value appears in the shared Flexcomm Interface peripheral ID register when USART is selected.	0xE0102100	34.6.20

34.6.1 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: Only the CFG register can be written when the ENABLE bit = 0. CFG can be set up by software with ENABLE = 1, then the rest of the USART can be configured.

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 647. USART Configuration register (CFG, offset 0x000)

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. When re-enabled, the USART will immediately be ready to transmit 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.	-
3:2	DATALEN		Selects the data size for the USART.	0
		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.	

Table 647. USART Configuration register (CFG, offset 0x000) *…continued*

Bit	Symbol	Value	Description	Reset Value
5:4	PARITYSEL		Selects what type of parity is used by the USART.	0
		0x0	No parity.	
		0x1	Reserved.	
		0x2	Even parity. Adds a bit to each character such that the number of 1s in a transmitted character is even, and the number of 1s in a received character is expected to be even.	
		0x3	Odd parity. Adds a bit to each character such that the number of 1s in a transmitted character is odd, and the number of 1s in a received character is expected to be odd.	
6	STOPLEN		Number of stop bits appended to transmitted data. Only a single stop bit is required for received data.	0
		0	1 stop bit.	
		1	2 stop bits. This setting should only be used for asynchronous communication.	
7	MODE32K		Selects standard or 32 kHz clocking mode.	0
		0	Disabled. USART uses standard clocking.	
		1	Enabled. USART uses the 32 kHz clock from the RTC oscillator as the clock source to the BRG, and uses a special bit clocking scheme.	
8	LINMODE		LIN break mode enable.	0
		0	Disabled. Break detect and generate is configured for normal operation.	
		1	Enabled. Break detect and generate is configured for LIN bus operation.	
9	CTSEN		CTS Enable. Determines whether CTS is used for flow control. CTS can be from the input pin, or from the USART's own RTS if loopback mode is enabled.	0
		0	No flow control. The transmitter does not receive any automatic flow control signal.	
		1	Flow control enabled. The transmitter uses the CTS input (or RTS output in loopback mode) for flow control purposes.	
10	-	-	Reserved. Read value is undefined, only zero should be written.	-
11	SYNCEN		Selects synchronous or asynchronous operation.	0
		0	Asynchronous mode.	
		1	Synchronous mode.	
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.	-
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.	

Table 647. USART Configuration register (CFG, offset 0x000) *…continued*

Bit	Symbol	Value	Description	Reset Value
17:16	-	-	Reserved. Read value is undefined, only zero should be written.	-
18	OETA		Output Enable Turnaround time enable for RS-485 operation.	0
		0	Disabled. If selected by OESEL, the Output Enable signal deasserted at the end of the last stop bit of a transmission.	
		1	Enabled. If selected by OESEL, the Output Enable signal remains asserted for one character time after the end of 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	Standard. The RTS signal is used as the standard flow control function.	
		1	RS-485. The RTS signal configured 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	Standard. 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 USART. 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	Standard. 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 USART 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.	-

34.6.2 USART control register

The CTL register controls aspects of USART operation that are more likely to change during operation.

Table 648. USART Control register (CTL, offset 0x004)

Bit	Symbol	Value	Description	Reset Value
0	-	-	Reserved. Read value is undefined, only zero should be written.	-
1	TXBRKEN		Break Enable.	0
		0	Normal operation.	
		1	Continuous break. 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.	-
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.	-
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. 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.	-

Table 648. USART Control register (CTL, offset 0x004) *…continued*

Bit	Symbol	Value	Description	Reset Value
16	AUTOBAUD		Autobaud enable.	0
		0	Disabled. USART is in normal operating mode.	
		1	Enabled. USART is in autobaud mode. This bit should only be set when the USART 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 AERR.	
31:17	-	-	Reserved. Read value is undefined, only zero should be written.	-

34.6.3 USART status register

The STAT register primarily provides a set of USART status flags (not including FIFO status) 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 651.

The error flags for received noise, parity error, and framing error are set immediately upon detection and remain set until cleared by software action in STAT.

Table 649. USART status register (STAT, offset 0x008)

Bit	Symbol	Description	Reset value	Access [1]
0	-	Reserved. Read value is undefined, only zero should be written.	-	-
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	-	Reserved. Read value is undefined, only zero should be written.	-	-
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	R/W1C
6	TXDISSTAT	Transmitter disabled status flag. When 1, this bit indicates that the USART transmitter is fully idle after being disabled via the TXDIS bit in the CFG register (TXDIS = 1).	0	RO
9:7	-	Reserved. Read value is undefined, only zero should be written.	-	-
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	R/W1C
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 mode immediately when a start is detected. Cleared by software.	0	R/W1C

Table 649. USART status register (STAT, offset 0x008) *…continued*

Bit	Symbol	Description	Reset value	Access [1]
13	FRAMERRINT	Framing Error interrupt flag. This flag is set when a character is received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.	0	R/W1C
14	PARITYERRINT	Parity Error interrupt flag. This flag is set when a parity error is detected in a received character.	0	R/W1C
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	R/W1C
16	ABERR	Auto baud error. An auto baud error can occur if the BRG counts to its limit before the end of the start bit that is being measured, essentially an auto baud time-out.	0	R/W1C
31:17	-	Reserved. Read value is undefined, only zero should be written.	-	-

[1] RO = Read-Only, R/W1C = Write 1 to Clear.

34.6.4 USART interrupt enable read and set register

The INTENSET register is used to enable various USART interrupt sources (not including FIFO interrupts). Enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. Interrupt enables may also be read back 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 650. USART interrupt enable read and set register (INTENSET, offset 0x00C)

Bit	Symbol	Description	Reset Value
2:0	-	Reserved. Read value is undefined, only zero should be written.	-
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.	-
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
10:7	-	Reserved. Read value is undefined, only zero should be written.	-
11	DELTARXBRKEN	When 1, enables an interrupt when a change of state has occurred in the detection of a received break condition (break condition asserted or deasserted).	0
12	STARTEN	When 1, enables an interrupt when a received start bit has been detected.	0
13	FRAMERREN	When 1, enables an interrupt when a framing error has been detected.	0
14	PARITYERREN	When 1, enables an interrupt when a parity error has been detected.	0
15	RXNOISEEN	When 1, enables an interrupt when noise is detected. See description of the RXNOISEINT bit in Table 649.	0
16	ABERREN	When 1, enables an interrupt when an auto baud error occurs.	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	-

34.6.5 USART interrupt enable clear register

The INTENCLR register is used to clear bits in the INTENSET register.

Table 651. USART interrupt enable clear register (INTENCLR, offset 0x010)

Bit	Symbol	Description	Reset value
2:0	-	Reserved. Read value is undefined, only zero should be written.	-
3	TXIDLECLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
4	-	Reserved. Read value is undefined, only zero should be written.	-
5	DELTACTSCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
6	TXDISCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
10:7	-	Reserved. Read value is undefined, only zero should be written.	-
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.	-

34.6.6 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 Flexcomm Interface clock (FCLK) 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.

For more information on USART clocking, see Section 34.7.2 “Clocking and baud rates” and Section 34.3.1 “Configure the Flexcomm Interface clock and USART baud rate”.

Remark: To change a baud rate after a USART is running, 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 BRGVAL.
4. Write to the CFG register to set the enable bit to 1.

Table 652. USART Baud Rate Generator register (BRG, offset 0x020)

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 = FCLK is used directly by the USART function. 1 = FCLK is divided by 2 before use by the USART function. 2 = FCLK is divided by 3 before use by the USART function. ... 0xFFFF = FCLK is divided by 65,536 before use by the USART function.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

34.6.7 USART interrupt status register

The read-only INTSTAT register provides a view of those interrupt flags that are currently enabled. It can simplify software handling of interrupts. See Table 649 for detailed descriptions of the interrupt flags.

Table 653. USART interrupt status register (INTSTAT, offset 0x024)

Bit	Symbol	Description	Reset value
2:0	-	Reserved. Read value is undefined, only zero should be written.	-
3	TXIDLE	Transmitter Idle status.	0
4	-	Reserved. Read value is undefined, only zero should be written.	-
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
10:7	-	Reserved. Read value is undefined, only zero should be written.	-
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	ABERRINT	Auto baud Error Interrupt flag.	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	-

34.6.8 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 function clock rate is not a multiple of 16 times the expected maximum baud rate. For all modes where the OSR setting is used, the USART 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 654. Oversample selection register (OSR, offset 0x028)

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

34.6.9 Address register

The ADDR register holds the address for hardware address matching in address detect mode with automatic address matching enabled.

Table 655. Address register (ADDR, offset 0x02C)

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.	-

34.6.10 FIFO Configuration register

This register configures FIFO usage. A peripheral function within the Flexcomm Interface must be selected prior to configuring the FIFO.

Table 656. FIFO Configuration register (FIFOCFG - offset 0xE00)

Bit	Symbol	Value	Description	Reset value	Access
0	ENABLETX		Enable the transmit FIFO.	0	R/W
		0	The transmit FIFO is not enabled.		
		1	The transmit FIFO is enabled.		
1	ENABLERX		Enable the receive FIFO.	0	R/W
		0	The receive FIFO is not enabled.		
		1	The receive FIFO is enabled.		
3:2	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
5:4	SIZE		FIFO size configuration. This is a read-only field. 0x0 = FIFO is configured as 16 entries of 8 bits. 0x1, 0x2, 0x3 = not applicable to USART.	NA	RO
11:6	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
12	DMATX		DMA configuration for transmit.	0	R/W
		0	DMA is not used for the transmit function.		
		1	Generate a DMA request for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.		
13	DMARX		DMA configuration for receive.	0	R/W
		0	DMA is not used for the receive function.		
		1	Generate a DMA request for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.		

Table 656. FIFO Configuration register (FIFOCFG - offset 0xE00) *...continued*

Bit	Symbol	Value	Description	Reset value	Access
14	WAKETX		Wake-up for transmit FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.	0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.		
15	WAKERX		Wake-up for receive FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.	0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.		
16	EMPTYTX	-	Empty command for the transmit FIFO. When a 1 is written to this bit, the TX FIFO is emptied.	-	WO
17	EMPTYRX	-	Empty command for the receive FIFO. When a 1 is written to this bit, the RX FIFO is emptied.	-	WO
31:18	-	-	Reserved. Read value is undefined, only zero should be written.	-	-

34.6.11 FIFO status register

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Table 657. FIFO status register (FIFOSTAT - offset 0xE04)

Bit	Symbol	Description	Access	Reset value
0	TXERR	TX FIFO error. Will be set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit.	R/W1C	0
1	RXERR	RX FIFO error. Will be set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit.	R/W1C	0
2	-	Reserved. Read value is undefined, only zero should be written.	-	-
3	PERINT	Peripheral interrupt. When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral's STAT register.	RO	0

Table 657. FIFO status register (FIFOSTAT - offset 0xE04) *…continued*

Bit	Symbol	Description	Access	Reset value
4	TXEMPTY	Transmit FIFO empty. When 1, the transmit FIFO is empty. The peripheral may still be processing the last piece of data.	RO	1
5	TXNOTFULL	Transmit FIFO not full. When 1, the transmit FIFO is not full, so more data can be written. When 0, the transmit FIFO is full and another write would cause it to overflow.	RO	1
6	RXNOTEMPTY	Receive FIFO not empty. When 1, the receive FIFO is not empty, so data can be read. When 0, the receive FIFO is empty.	RO	0
7	RXFULL	Receive FIFO full. When 1, the receive FIFO is full. Data needs to be read out to prevent the peripheral from causing an overflow.	RO	0
12:8	TXLVL	Transmit FIFO current level. A 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. Other values tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, the TXEMPTY and TXNOTFULL flags will be 0.	RO	0
15:13	-	Reserved. Read value is undefined, only zero should be written.	-	-
20:16	RXLVL	Receive FIFO current level. A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. Other values tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, the RXFULL and RXNOTEMPTY flags will be 1.	RO	0
31:21	-	Reserved. Read value is undefined, only zero should be written.	-	-

34.6.12 FIFO trigger level settings register

This register allows selecting when FIFO-level related interrupts occur.

Table 658. FIFO trigger level settings register (FIFOTRIG - offset 0xE08)

Bit	Symbol	Value	Description	Reset value
0	TXLVLENA		Transmit FIFO level trigger enable. The FIFO level trigger will cause an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMATX in FIFOCFG).	0
		0	Transmit FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.	
1	RXLVLENA		Receive FIFO level trigger enable. This trigger will become an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMARX in FIFOCFG).	0
		0	Receive FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.	
7:2	-	-	Reserved. Read value is undefined, only zero should be written.	-
11:8	TXLVL		Transmit FIFO level trigger point. This field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. 0 = generate an interrupt when the TX FIFO becomes empty. 1 = generate an interrupt when the TX FIFO level decreases to one entry. ... 15 = generate an interrupt when the TX FIFO level decreases to 15 entries (is no longer full).	0

Table 658. FIFO trigger level settings register (FIFOTRIG - offset 0xE08) *...continued*

Bit	Symbol	Value	Description	Reset value
15:12	-	-	Reserved. Read value is undefined, only zero should be written.	-
19:16	RXLVL		Receive FIFO level trigger point. The RX FIFO level is checked when a new piece of data is received. This field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. 0 = generate an interrupt when the RX FIFO has one entry (is no longer empty). 1 = generate an interrupt when the RX FIFO has two entries. ... 15 = generate an interrupt when the RX FIFO increases to 16 entries (has become full).	0
31:20	-	-	Reserved. Read value is undefined, only zero should be written.	-

34.6.13 FIFO interrupt enable set and read

This register is used to enable various interrupt sources. 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 FIFOINTENCLR register is used to clear bits in this register.

Table 659. FIFO interrupt enable set and read register (FIFOINTENSET - offset 0xE10)

Bit	Symbol	Value	Description	Reset value
0	TXERR		Determines whether an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register.	0
		0	No interrupt will be generated for a transmit error.	
		1	An interrupt will be generated when a transmit error occurs.	
1	RXERR		Determines whether an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register.	0
		0	No interrupt will be generated for a receive error.	
		1	An interrupt will be generated when a receive error occurs.	
2	TXLVL		Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0
		0	No interrupt will be generated based on the TX FIFO level.	
		1	If TXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.	
3	RXLVL		Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0
		0	No interrupt will be generated based on the RX FIFO level.	
		1	If RXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.	
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

34.6.14 FIFO interrupt enable clear and read

The FIFOINTENCLR register is used to clear interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Table 660. FIFO interrupt enable clear and read (FIFOINTENCLR - offset 0xE14)

Bit	Symbol	Description	Reset value
0	TXERR	Writing a one to this bit disables the TXERR interrupt.	0x0
1	RXERR	Writing a one to this bit disables the RXERR interrupt.	0x0
2	TXLVL	Writing a one to this bit disables the interrupt caused by the transmit FIFO reaching the level specified by the TXLVL field in the FIFOTRIG register.	0x0
3	RXLVL	Writing a one to this bit disables the interrupt caused by the receive FIFO reaching the level specified by the RXLVL field in the FIFOTRIG register.	0x0
31:4	-	Reserved. Read value is undefined, only zero should be written.	-

34.6.15 FIFO interrupt status register

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are both pending and currently enabled. This can simplify software handling of interrupts. Refer to the descriptions of interrupts in Section 34.6.11 “FIFO status register” and Section 34.6.12 “FIFO trigger level settings register” for details.

Table 661. FIFO interrupt status register (FIFOINTSTAT - offset 0xE18)

Bit	Symbol	Description	Reset value
0	TXERR	TX FIFO error.	0
1	RXERR	RX FIFO error.	0
2	TXLVL	Transmit FIFO level interrupt.	0
3	RXLVL	Receive FIFO level interrupt.	0
4	PERINT	Peripheral interrupt.	0
31:5	-	Reserved. Read value is undefined, only zero should be written.	-

34.6.16 FIFO write data register

The FIFOWR register is used to write values to be transmitted to the FIFO.

Table 662. FIFO write data register (FIFOWR - offset 0xE20)

Bit	Symbol	Description	Reset value
8:0	TXDATA	Transmit data to the FIFO.	NA

34.6.17 FIFO read data register

The FIFORD register is used to read values that have been received by the FIFO.

Table 663. FIFO read data register (FIFORD - offset 0xE30)

Bit	Symbol	Description	Reset value
8:0	RXDATA	Received data from the FIFO. The number of bits used depends on the DATALEN and PARITYSEL settings.	NA
12:9	-	Reserved, the value read from a reserved bit is not defined.	-
13	FRAMERR	Framing Error status flag. This bit reflects the status for the data it is read along with from the FIFO, and indicates that the character 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.	NA

Table 663. FIFO read data register (FIFORD - offset 0xE30) *…continued*

Bit	Symbol	Description	Reset value
14	PARITYERR	Parity Error status flag. This bit reflects the status for the data it is read along with from the FIFO. This bit will be set when a parity error is detected in a received character.	NA
15	RXNOISE	Received Noise flag. See description of the RxNoiseInt bit in Table 649.	NA
31:16	-	Reserved, the value read from a reserved bit is not defined.	-

34.6.18 FIFO data read with no FIFO pop

This register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (that is, leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Table 664. FIFO data read with no FIFO pop (FIFORDNOPOP - offset 0xE40)

Bit	Symbol	Description	Reset value
8:0	RXDATA	Received data from the FIFO.	NA
12:9	-	Reserved, the value read from a reserved bit is not defined.	-
13	FRAMERR	Framing Error status flag.	NA
14	PARITYERR	Parity Error status flag.	NA
15	RXNOISE	Received Noise flag.	NA
31:16	-	Reserved, the value read from a reserved bit is not defined.	-

34.6.19 FIFO size register

The FIFOSIZE register provides the size FIFO for the selected Flexcomm function on this device.

Table 665. FIFO size register (FIFOSIZE - offset = 0xE48)

Bit	Symbol	Description	Reset value
4:0	FIFOSIZE	Provides the size of the FIFO for software. The size for the USART FIFO is 16 entries.	0x10
31:5	-	Reserved.	-

34.6.20 Module identification register

The ID register identifies the type and revision of the USART module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 666. Module identification register (ID - offset 0xFFC)

Bit	Symbol	Description	Reset Value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x0
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE010

34.7 Functional description

34.7.1 AHB bus access

The bus interface to the USART registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the USART function.

34.7.2 Clocking and baud rates

In order to use the USART, clocking details must be defined such as setting up the clock source selection, the BRG, and setting up the FRG if it is the selected clock source.

Also see Section 34.3.1 “Configure the Flexcomm Interface clock and USART baud rate”.

34.7.2.1 Fractional Rate Generator (FRG)

The Fractional Rate Generator can be used to obtain more precise baud rates when the function 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 Fractional Rate Generator, which provides the base clock that may be used by any Flexcomm Interface. 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 input 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 USARTs 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, see Section 4.5.49 “Fractional rate divider for each Flexcomm Interface frequency”.

34.7.2.2 Baud Rate Generator (BRG)

The Baud Rate Generator, see Section 34.6.6 “USART baud rate generator register” 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.

34.7.2.3 32 kHz mode

In order to use a 32 kHz clock 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 integer 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 USART 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.

34.7.3 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 and FIFO level triggering 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.

34.7.4 Synchronous mode

In synchronous mode, a master generates a clock as defined by the clock selection and BRG, which is used to transmit and receive data. As a slave, the external clock is used to transmit and receive data. There is no overclocking in either case.

34.7.5 Flow control

The USART supports both hardware and software flow control.

34.7.5.1 Hardware flow control

The USART supports hardware flow control using RTS and/or CTS signalling. If RTS is configured to appear on a device pin so that it can be sent to an external device, it indicates to an external device the ability of the receiver to receive more data. It can also be used internally to throttle the transmitter from the receiver, which can be especially useful if loopback mode is enabled.

If connected to a pin, and if enabled to do so, the CTS input can allow an external device to throttle the USART transmitter. Both internal and external CTS can be used separately or together.

Figure 119 shows an overview of RTS and CTS within the USART.

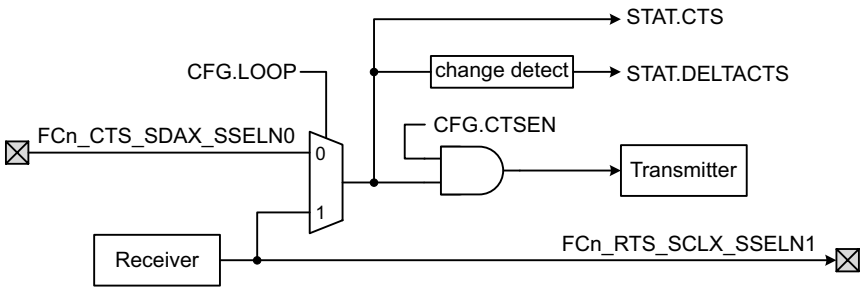


Fig 119. Hardware flow control using RTS and CTS

34.7.5.2 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).

34.7.6 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. Before an autobaud operation is requested, the BRG value must be set to 0. 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 data will become available in the FIFO or ABERR will be asserted at some point, at which time software should turn off autobaud.

Autobaud has no meaning and should not be enabled when the USART is in synchronous mode.

34.7.7 RS-485 support

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

This USART has provisions for hardware address recognition (see the AUTOADDR bit in the CFG register in Section 34.6.1 “USART configuration register” and the ADDR register in Section 34.6.9 “Address register”), as well as software address recognition (see the ADDRDET bit in the CTL register in Section 34.6.2 “USART control register”).

Automatic data direction control with the RTS pin can be set up using the OESEL, OEPOL, and OETA bits in the CFG register (Section 34.6.1 “USART configuration register”). Data direction control can also be implemented in software using a GPIO pin.

34.7.8 Oversampling

Typical industry standard USARTs 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 USART 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 function clock is very low. For example, the closest actual rate near 115,200 baud with a 12 MHz function 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 USART 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.

34.7.9 Break generation and detection

A line break may be sent at any time, regardless of other USART activity. Received break is also detected at any time, including during reception of a character. Received break is signaled when the RX input remains low for 16 bit times. Both the beginning and end of a received break are noted by the DELTARXBRK status flag, which can be used as an interrupt. See Section 34.7.10 “LIN bus” for details of LIN mode break.

In order to avoid corrupting any character currently being transmitted, it is recommended that the USART transmitter be disabled by setting the TXDIS bit in the CTL register, then waiting for the TXDISSTAT flag to be set prior to sending a break. Then a 1 may be written to the TXBRKEN bit in the CTL register. This sends a break until TXBRKEN is cleared, allowing any length break to be sent.

34.7.10 LIN bus

The only difference between standard operation and LIN mode is that LIN mode alters the way that break generation and detection is performed, see Section 34.7.9 “Break generation and detection” for details. When a break is requested by setting the TXBRKEN bit in the CTL register, then sending a dummy character, a 13 bit time break is sent. A received break is flagged when the RX input remains low for 11 bit times. As for non-LIN mode, a received character is also flagged, and accompanied by a framing error status.

As a LIN slave, the autobaud feature can be used to synchronize to a LIN sync byte, and will return the value of the sync byte as confirmation of success.

Wake-up for LIN can potentially be handled in a number of ways, depending on the system, and what clocks may be running in a slave device. For instance, as long as the USART is receiving internal clocks allowing it to function, it can be set to wake up the CPU for any interrupt, including a received start bit. If there are no clocks running, the GPIO function of the USART RX pin can be programmed to wake up the device.

35.1 How to read this chapter

SPI functions are available on all LPC55S6x/LPC55S2x/LPC552x devices as a selectable function in each Flexcomm Interface. Up to eight Flexcomm Interfaces and one high-speed Flexcomm Interface (Flexcomm Interface 8) are available.

35.2 Features

- Master and slave operation.
- Data transmits of 4 to 16 bits supported directly. Larger frames supported by software.
- The SPI function supports separate transmit and receive FIFOs with eight entries each.
- Supports DMA transfers: SPIn transmit and receive functions can operated with the system DMA controller.
- Data can be transmitted to a slave without the need to read incoming data. This can be useful while setting up an SPI memory.
- Up to four slave select input/outputs with selectable polarity and flexible usage.

35.3 Basic configuration

Initial configuration of an SPI peripheral is accomplished as follows:

- If needed, use the PRESETCTRL1 or PRESETCTRL2 register, see Table 114 to reset the Flexcomm Interface that is about to have a specific peripheral function selected
- Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface (Flexcomm Interface Section 32.7.1 “Peripheral Select and Flexcomm Interface ID register”.
- Configure the FIFOs for operation.
- Configure the SPI for receiving and transmitting data
  - In the AHBCLKCTRL1 or AHBCLKCTRL2 (Table 56) register, set the appropriate bit for the related Flexcomm Interface to enable the clock to the register interface
  - Enable or disable the related Flexcomm Interface interrupts in the NVIC (Table 72).
  - Configure the required Flexcomm Interface pin functions through IOCON. See Section 35.4 “Pin description”
  - Configure the Flexcomm Interface clock and SPI data rate. See.Section 35.7.4 “Clocking and date rates”
  - Set the RXIGNORE bit to only transmit data and not read the incoming data. Otherwise, the transmit halts when the FIFORD buffer is full.

- For a slave, potentially set the TXIGNORE bit to only receive data.

The Flexcomm interface function clock frequency (at the output of the FRG associated with the flexcomm) shall not be above **100 MHz**.

The SPI peripheral clock (the SPI protocol Bus clock) shall not be above **33 MHz in Master Mode** or above **16 MHz in Slave Mode**

The **High-Speed Flexcomm** interface function clock frequency (at the output of the FRG associated with the flexcomm) shall not be above **100 MHz**.

The High-Speed SPI peripheral clock (the SPI protocol Bus clock) shall not be above **50 MHz in Master Mode** or above **25 MHz in Slave Mode**

- Configure the SPI function to wake up the part from low power modes. See Section 35.3.1 “Configure the SPI for wake-up”.

35.3.1 Configure the SPI for wake-up

In sleep-mode, any signal that triggers an SPI interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the SPI clock is configured to be active in sleep-mode, the SPI can wake up the part independently of whether the SPI block is configured in master or slave mode.

In deep-sleep mode, the SPI clock is turned off. However, if the SPI is configured in slave mode and an external master on the provides the clock signal, the SPI can create an interrupt asynchronously and wake up the device. The appropriate interrupt(s) must be enabled in the SPI and the NVIC.

35.3.1.1 Wake-up from sleep-mode

- Configure the SPI in either master or slave mode. See Table 670.
- Enable the SPI interrupt in the NVIC.
- Any enabled SPI interrupt wakes up the part from sleep-mode.

35.3.1.2 Wake-up from deep-sleep mode

- Configure the SPI in slave mode. See Table 670. The SCK function must be connected to a pin and the pin connected to the master
- Enable the SPI interrupt as wake-up source using the POWER_EnterDeepSleep low power API.
- Enable the SPI interrupt in the NVIC.
- Enable desired SPI interrupts. Examples are the following wake-up events:
  - A change in the state of the SSEL pins.
  - Data available to be received.
  - Receive FIFO overflow.

35.4 Pin description

The SPI signals are movable Flexcomm Interface functions and are assigned to external pins via IOCON. See Chapter 11 “LPC5500 I/O pin configuration (IOCON)”. Recommended IOCON settings are shown in Table 668.

Table 667. SPI pin description

Function	Type	Pin name used in pin description chapter	Description
SCK	I/O	FCn_SCK or HS_SPI_SCK	Serial Clock for SPI on Flexcomm Interface n. SCK is a clock signal used to synchronize the transfer of data. It is driven by the master and received by the slave. When the SPI interface is used, the clock is programmable to be active-high or active-low. SCK only switches during a data transfer. It is driven whenever the master bit in CFG equals 1, regardless of the state of the enable bit.
MOSI	I/O	FCn_RXD_SDA_MOSI_DATA or HS_SPI_MOSI	Master Out Slave. In for SPI on Flexcomm Interface n. The MOSI signal transfers serial data from the master to the slave. When the SPI is a master, it outputs serial data on this signal. When the SPI is a slave, it clocks in serial data from this signal. MOSI is driven whenever the master bit in CFG equals 1, regardless of the state of the enable bit.
MISO	I/O	FCn_TXD_SCL_MISO_WS or HS_SPI_MISO	Master In Slave Out for SPI on Flexcomm Interface n. The MISO signal transfers serial data from the slave to the master. When the SPI is a master, serial data is input from this signal. When the SPI is a slave, serial data is output to this signal. MISO is driven when the SPI block is enabled, the master bit in CFG equals 0, and when the slave is selected by one or more SSEL signals.
SSEL0	I/O	FCn_CTS_SDA_SSEL0 or HS_SPI_SSEL0	Slave select 0 for SPI on Flexcomm Interface n. When the SPI interface is a master, it will drive the SSEL signals to an active state before the start of serial data and then release them to an inactive state after the serial data has been sent. By default, this signal is active low but can be selected to operate as active high. When the SPI is a slave, any SSEL in an active state indicates that this slave is being addressed. The SSEL pin is driven whenever the master bit in the CFG register equals 1, regardless of the state of the enable bit.
SSEL1	I/O	FCn_RTS_SCL_SSEL1 or HS_SPI_SSEL1	Slave select 1 for SPI on Flexcomm Interface n.
SSEL2	I/O	FCn_SSEL2 or HS_SPI_SSEL2	Slave select 2 for SPI on Flexcomm Interface n.
SSEL3	I/O	FCn_SSEL3 or HS_SPI_SSEL3	Slave select 3 for SPI on Flexcomm Interface n.

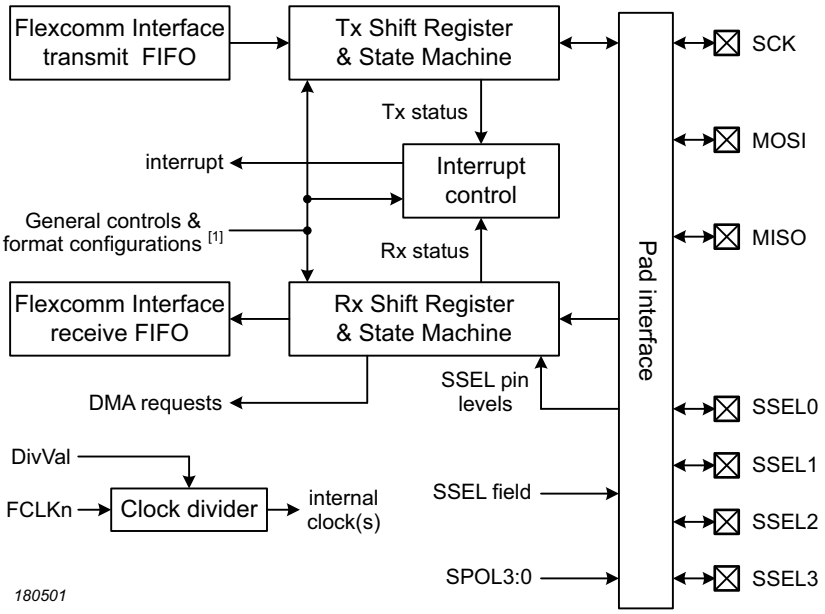
Table 668. Suggested SPI pin settings

IOCON bit(s)	Type D pin	Type A pin (GPIO)	Type I pin (I^2C)
31:16	Reserved	Reserved	Reserved
15	Reserved	Reserved	0b - Enabled. I^2C 50 ns glitch filter enabled. 1b - Disabled. I^2C 50 ns glitch filter disabled.
14	Reserved	Reserved	Select GPIO/I2C mode. Generally set to 1
13	Reserved	Reserved	0b - Enabled. Pull resistor is connected. 1b - Disabled. IO is in open drain.
12	Reserved	Reserved	I^2C FILTER: 0 for fast / standard mode I^2C. 1 for fast mode plus or high-speed slave
11	Reserved	Reserved	**SSEL**: Generally set to 0.
10	Reserved. Set to 0	**ASW** enable: Analog switch input control. Usable only if DIGIMODE = 0b0. Generally set to 0.	Reserved. Set to 0
9	**OD: Controls open-drain mode.** 0b - Normal. Normal push-pull output. 1b - Open-drain. Simulated open-drain output (high drive disabled). Generally Set to 0 unless open drain is desired	Same as type D	Same as type D
8	**DIGIMODE**: 0b - Analog mode, digital input is disabled. 1b - Digital mode, digital input is enabled. Generally set to 1.	Same as type D	Same as type D
7	**INVERT**: **Input polarity.** 0b - Disabled. Input function is not inverted. 1b - Enabled. Input is function inverted. Generally set to 0	Same as type D	Same as type D

Table 668. Suggested SPI pin settings *…continued*

IOCON bit(s)	Type D pin	Type A pin (GPIO)	Type I pin (I^2C)
6	**SLEW**, **Driver slew rate.** 0b - Standard mode, output slew rate control is enabled. More outputs can be switched simultaneously. 1b - Fast mode, slew rate control is disabled. Refer to the appropriate specific device data sheet for details. Generally set to 0	Same as type D	Same as type D
5:4	**MODE: Selects function mode (on-chip pull-up/pull-down resistor control).** 00b - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01b - Pull-down. Pull-down resistor enabled. 10b - Pull-up. Pull-up resistor enabled. 11b - Repeater. Repeater mode. Generally set to 0	Same as type D	Same as type D
3:0	**FUNC: Selects pin function.** 0000b - Alternative connection 0. 0001b - Alternative connection 1. 0010b - Alternative connection 2. 0011b - Alternative connection 3. 0100b - Alternative connection 4. 0101b - Alternative connection 5. 0110b - Alternative connection 6. 0111b - Alternative connection 7.	Same as type D	Same as type D

35.5 General description



(1) Includes CPOL. CPHA, LDBF, LEN, MASTER, ENABLE, TRANSFER_DELAY, FRAME_DELAY, PRE_DELAY, POST_DELAY, SOT, EOT, EOF, RXIGNORE, TXIGNORE, individual enables.

Fig 120. SPI block diagram

35.6 Register description

Address offsets are within the address space of the related Flexcomm Interface. The reset value reflects the data stored in used bits only. It does not include reserved bits content.

35.6.1 FLEXCOMM memory map

SPI0 base address: 4008_6000h

SPI1 base address: 4008_7000h

SPI2 base address: 4008_8000h

SPI3 base address: 4008_9000h

SPI4 base address: 4008_A000h

SPI5 base address: 4009_6000h

SPI6 base address: 4009_7000h

SPI7 base address: 4009_8000h

SPI8 base address: 4009_F000h (HS_SPI)

Table 669. SPI register overview

Name	Access	Offset	Description	Reset value	Section
Registers for the SPI function:					
CFG	R/W	0x400	SPI configuration register.	0	35.6.2
DLY	R/W	0x404	SPI delay register.	0	35.6.3
STAT	R/W	0x408	SPI status. Some status flags can be cleared by writing a 1 to that bit position.	-	35.6.4
INTENSET	R/W	0x40C	SPI interrupt enable read and set. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set.	0	35.6.5
INTENCLR	WO	0x410	SPI interrupt enable clear. Writing a 1 to any implemented bit position causes the corresponding bit in INTENSET to be cleared.	-	35.6.6
DIV	R/W	0x424	SPI clock divider.	0	35.6.7
INTSTAT	RO	0x428	SPI interrupt status.	-	35.6.8
Registers for FIFO control and data access:					
FIFOCFG	R/W	0xE00	FIFO configuration and enable register.	0x13	35.6.9
FIFOSTAT	R/W	0xE04	FIFO status register.	0x30	35.6.10
FIFOTRIG	R/W	0xE08	FIFO trigger level settings for interrupt and DMA request.	0	35.6.11
FIFOINTENSET	R/W1S	0xE10	FIFO interrupt enable set (enable) and read register.	0	35.6.12
FIFOINTENCLR	R/W1C	0xE14	FIFO interrupt enable clear (disable) and read register.	0	35.6.13
FIFOINTSTAT	RO	0xE18	FIFO interrupt status register.	0	35.6.14
FIFOWR	WO	0xE20	FIFO write data.	-	35.6.15
FIFORD	RO	0xE30	FIFO read data.	-	35.6.16
FIFORDNOPOP	RO	0xE40	FIFO data read with no FIFO pop.	-	35.6.17
FIFOSIZE	R	0xE48	FIFO size.	0x8	
ID register:					
ID	RO	0xFFC	SPI module identification. This value appears in the shared Flexcomm Interface peripheral ID register when SPI is selected.	0xE0201200	35.6.19

35.6.2 SPI configuration register

The CFG register contains information for the general configuration of the SPI. Typically, this information is not changed during operation. See Table 672 for the description of the master idle status.

Remark: A setup sequence is recommended for initial SPI setup (after the SPI function is selected, see Chapter 32 “LPC55S6x/LPC55S2x/LPC552x Flexcomm Interface Serial Communication”, and when changes need to be made to settings in the CFG register after the interface is in use. See the list below. In the case of changing existing settings, the interface should first be disabled by clearing the ENABLE bit once the interface is fully idle. See Table 672 for the description of the master idle status (MSTIDLE).

- Disable the FIFO by clearing the ENABLETX and ENABLERX bits in FIFOCFG.
- Setup the SPI interface in the CFG register, leaving ENABLE = 0.
- Enable the FIFO by setting the ENABLETX and/or ENABLERX bits in FIFOCFG.

- Enable the SPI by setting the ENABLE bit in CFG.

Table 670. SPI configuration register (CFG, offset 0x400)

Bit	Symbol	Value	Description	Reset value
0	ENABLE		SPI enable.	0
		0	Disabled. The SPI is disabled and the internal state machine and counters are reset.	
		1	Enabled. The SPI is enabled for operation.	
1	-	-	Reserved. Read value is undefined, only zero should be written.	-
2	MASTER		Master mode select.	0
		0	Slave mode. The SPI will operate in slave mode. SCK, MOSI, and the SSEL signals are inputs, MISO is an output.	
		1	Master mode. The SPI will operate in master mode. SCK, MOSI, and the SSEL signals are outputs, MISO is an input.	
3	LSBF		LSB first mode enable.	0
		0	Standard. Data is transmitted and received in standard MSB first order.	
		1	Reverse. Data is transmitted and received in reverse order (LSB first).	
4	CPHA		Clock phase select.	0
		0	Change. The SPI captures serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge.	
		1	Capture. The SPI changes serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge.	
5	CPOL		Clock polarity select.	0
		0	Low. The rest state of the clock (between transfers) is low.	
		1	High. The rest state of the clock (between transfers) is high.	
6	-	-	Reserved. Read value is undefined, only zero should be written.	-
7	LOOP		Loop back mode enable. Loop back mode applies only to master mode, and connects transmit and receive data connected together to allow simple software testing.	0
		0	Disabled.	
		1	Enabled.	
8	SPOL0		SSEL0 polarity select.	0
		0	Low. The SSEL0 pin is active low.	
		1	High. The SSEL0 pin is active high.	
9	SPOL1		SSEL1 polarity select.	0
		0	Low. The SSEL1 pin is active low.	
		1	High. The SSEL1 pin is active high.	
10	SPOL2		SSEL2 polarity select.	0
		0	Low. The SSEL2 pin is active low.	
		1	High. The SSEL2 pin is active high.	

Table 670. SPI configuration register (CFG, offset 0x400) *…continued*

Bit	Symbol	Value	Description	Reset value
11	SPOL3		SSEL3 polarity select.	0
		0	Low. The SSEL3 pin is active low.	
		1	High. The SSEL3 pin is active high.	
31:12	-	-	Reserved. Read value is undefined, only zero should be written.	-

35.6.3 SPI delay register

The DLY register controls several programmable delays related to SPI signalling. These delays apply only to master mode, and are all stated in SPI clocks.

Timing details are shown in Section 35.7.3.1 “Pre_delay and Post_delay”, Section 35.7.3.2 “Frame_delay” and Section 35.7.3.3 “Transfer_delay”.

Table 671. SPI delay register (DLY, offset 0x404)

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

35.6.4 SPI status register

The STAT register provides SPI status flags for software to read, and a control bit for forcing an end of transfer. Flags other than read-only flags may be cleared by writing ones to corresponding bits of STAT.

In this register, the following notation is used: RO = read-only, W1C = write 1 to clear.

Table 672. SPI status register (STAT, offset 0x408)

Bit	Symbol	Description	Reset value	Access
3:0	-	Reserved. Read value is undefined, only zero should be written.	-	
4	SSA	Slave select assert. This flag is set whenever any slave select transitions from de-asserted to asserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become busy, and allows waking up the device from reduced power modes when a slave mode access begins. This flag is cleared by software.	0	W1C
5	SSD	Slave select de-assert. This flag is set whenever any asserted slave selects transition to de-asserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become idle. This flag is cleared by software.	0	W1C
6	STALLED	Stalled status flag. This indicates whether the SPI is currently in a stall condition.	0	RO
7	ENDTRANSFER	End Transfer control bit. Software can set this bit to force an end to the current transfer when the transmitter finishes any activity already in progress, as if the EOT flag had been set prior to the last transmission. This capability is included to support cases where it is not known when transmit data is written that it will be the end of a transfer. The bit is cleared when the transmitter becomes idle as the transfer comes to an end. Forcing an end of transfer in this manner causes any specified FRAME_DELAY and TRANSFER_DELAY to be inserted.	0	R/W1C
8	MSTIDLE	Master idle status flag. This bit is 1 whenever the SPI master function is fully idle. This means that the transmit holding register is empty and the transmitter is not in the process of sending data.	1	RO
31:9	-	Reserved. Read value is undefined, only zero should be written.	-	

[1] RO = read-only, W1C = write 1 to clear.

35.6.5 SPI interrupt enable read and set register

The INTENSET register is used to enable various SPI interrupt sources. Enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The INTENCLR register is used to clear bits in this register. See Table 672 for details of the interrupts.

Table 673. SPI interrupt enable read and set register (INTENSET, offset = 0x40C)

Bit	Symbol	Value	Description	Reset value
3:0	-	-	Reserved. Read value is undefined, only zero should be written.	-
4	SSAEN		Slave select assert interrupt enable. Determines whether an interrupt occurs when the slave select is asserted.	0
		0	Disabled. No interrupt will be generated when any slave select transitions from de-asserted to asserted.	
		1	Enabled. An interrupt will be generated when any slave select transitions from de-asserted to asserted.	
5	SSDEN		Slave select de-assert interrupt enable. Determines whether an interrupt occurs when the slave select is de-asserted.	0
		0	Disabled. No interrupt will be generated when all asserted slave selects transition to de-asserted.	
		1	Enabled. An interrupt will be generated when all asserted slave selects transition to de-asserted.	
7:6	-	-	Reserved. Read value is undefined, only zero should be written.	
8	MSTIDLEEN		Master idle interrupt enable.	0
		0	No interrupt will be generated when the SPI master function is idle.	
		1	An interrupt will be generated when the SPI master function is fully idle.	
31:9	-	-	Reserved. Read value is undefined, only zero should be written.	-

35.6.6 SPI interrupt enable clear register

The INTENCLR register is used to clear interrupt enable bits in the INTENSET register.

Table 674. SPI interrupt enable clear register (INTENCLR, offset = 0x410)

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	-
4	SSAEN	Writing 1 clears the corresponding bit in the INTENSET register.	0
5	SSDEN	Writing 1 clears the corresponding bit in the INTENSET register.	0
7:6	-	Reserved. Read value is undefined, only zero should be written.	-
8	MSTIDLE	Writing 1 clears the corresponding bit in the INTENSET register	0
31:9	-	Reserved. Read value is undefined, only zero should be written.	-

35.6.7 SPI divider register

The DIV register determines the clock used by the SPI in master mode.

For details on clocking, see Section 35.7.4 “Clocking and date rates”

Table 675. SPI divider register (DIV, offset = 0x424)

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

35.6.8 SPI interrupt status register

The read-only INTSTAT register provides a view of the interrupt condition(s) that have occurred. Reading the register clears the bits. This can simplify software handling of interrupts. See Table 672 for detailed descriptions of the interrupt flags.

Table 676. SPI interrupt status register (INTSTAT, offset = 0x428)

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written	-
4	SSA	Slave select assert.	0
5	SSD	Slave select de-assert.	0
7:6	-	Reserved. Read value is undefined, only zero should be written.	-
8	MSTIDLE	Master idle status flag.	0
31:9	-	Reserved. Read value is undefined, only zero should be written.	-

35.6.9 FIFO configuration register

This register configures FIFO usage. A peripheral function within the Flexcomm Interface must be selected prior to configuring the FIFO.

Table 677. FIFO configuration register (FIFOCFG - offset = 0xE00)

Bit	Symbol	Value	Description	Reset value	Access
0	ENABLETX		Enable the transmit FIFO.	0	R/W
		0	The transmit FIFO is not enabled.		
		1	The transmit FIFO is enabled.		
1	ENABLERX		Enable the receive FIFO.	0	R/W
		0	The receive FIFO is not enabled.		
		1	The receive FIFO is enabled.		
3:2			Reserved. Read value is undefined, only zero should be written	-	
5:4	SIZE		FIFO size configuration. This is a read-only field. 0x1 = FIFO is configured as 8 entries of 16 bits. 0x0, 0x2, 0x3 = not applicable to SPI.	-	RO
11:6	-	-	Reserved. Read value is undefined, only zero should be written.	-	

Table 677. FIFO configuration register (FIFOCFG - offset = 0xE00) *…continued*

Bit	Symbol	Value	Description	Reset value	Access
12	DMATX		DMA configuration for transmit.	0	R/W
		0	DMA is not used for the transmit function.		
		1	Generate a DMA request for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.		
13	DMARX		DMA configuration for receive.	0	R/W
		0	DMA is not used for the receive function.		
		1	Generate a DMA request for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.		
14	WAKETX		Wake-up for transmit FIFO level. This allows the device to be woken from reduced power-modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.	0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.		
15	WAKERX		Wake-up for receive FIFO level. This allows the device to be woken from reduced power-modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.	0	R/W
		0	Only enabled interrupts will wake up the device form reduced power-modes.		
		1	A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.		
16	EMPTYTX	-	Empty command for the transmit FIFO. When a 1 is written to this bit, the TX FIFO is emptied.	-	WO
17	EMPTYRX		Empty command for the receive FIFO. When a 1 is written to this bit, the RX FIFO is emptied.	-	WO
31:18	-	-	Reserved. Read value is undefined, only zero should be written.	-	-

35.6.10 FIFO status register

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Table 678. FIFO status register (FIFOSTAT - offset = 0xE04)

Bit	Symbol	Description	Reset value	Access
0	TXERR	TX FIFO error. Will be set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit.	0	R/W1C
1	RXERR	RX FIFO error. Will be set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit.	0	R/W1C
2	-	Reserved. Read value is undefined, only zero should be written.	-	-
3	PERINT	Peripheral interrupt. When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral's STAT register.	0	RO
4	TXEMPTY	Transmit FIFO empty. When 1, the transmit FIFO is empty. The peripheral may still be processing the last piece of data.	1	RO
5	TXNOTFULL	Transmit FIFO not full. When 1, the transmit FIFO is not full, so more data can be written. When 0, the transmit FIFO is full and another write would cause it to overflow.	1	RO
6	RXNOTEMPTY	Receive FIFO not empty. When 1, the receive FIFO is not empty, so data can be read. When 0, the receive FIFO is empty.	0	RO
7	RXFULL	Receive FIFO full. When 1, the receive FIFO is full. Data needs to be read out to prevent the peripheral from causing an overflow.	0	RO
12:8	TXLVL	Transmit FIFO current level. A 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. Other values tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, the TXEMPTY and TXNOTFULL flags will be 0.	0	RO
15:13	-	Reserved. Read value is undefined, only zero should be written.	-	-
20:16	RXLVL	Receive FIFO current level. A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. Other values tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, the RXFULL and RXNOTEMPTY flags will be 1.	0	RO
31:21	-	Reserved. Read value is undefined, only zero should be written.	-	-

35.6.11 FIFO trigger setting register

This register allows selecting when FIFO-level related interrupts occur.

Table 679. FIFO trigger settings register (FIFOTRIG - offset = 0xE08)

Bit	Symbol	Value	Description	Reset value
0	TXLVLENA		Transmit FIFO level trigger enable. The TX FIFO level trigger will cause an interrupt if enabled in FIFOINTENSET,. This field is not used for DMA requests. See DMATX in Section 35.6.9 "FIFO configuration register"	0
		0	Transmit FIFO level does not generate a FIFO level trigger.	
		1	An trigger will be generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.	

Table 679. FIFO trigger settings register (FIFOTRIG - offset = 0xE08) *…continued*

Bit	Symbol	Value	Description	Reset value
1	RXLVLENA		Receive FIFO level trigger enable. The RX FIFO level trigger will cause an interrupt if enabled in FIFOINTENSET.This field is not used for DMA requests. See DMARX in Section 35.6.9 “FIFO configuration register”.	0
		0	Receive FIFO level does not generate a FIFO level trigger.	
		1	An trigger will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.	
10:2	-	-	Reserved. Read value is undefined, only zero should be written.	-
11:8	TXLVL		Transmit FIFO level trigger point. This field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. 0 = generate an interrupt when the TX FIFO becomes empty. 1 = generate an interrupt when the TX FIFO level decreases to one entry. ... 7 = generate an interrupt when the TX FIFO level decreases to 7 entries (is no longer full).	0
15:12	-	-	Reserved. Read value is undefined, only zero should be written.	
19:16	RXLVL		Receive FIFO level trigger point. The RX FIFO level is checked when a new piece of data is received. This field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. 0 = generate an interrupt when the RX FIFO has one entry (is no longer empty). 1 = generate an interrupt when the RX FIFO has two entries. ... 7 = generate an interrupt when the RX FIFO has 8 entries (has become full).	0
31:20	-		Reserved. Read value is undefined, only zero should be written.	-

35.6.12 FIFO interrupt enable set and read register

This register is used to enable various interrupt sources. 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 FIFOINTENCLR register is used to clear bits in this register.

Table 680. FIFO interrupt enable set and read register (FIFOINTENSET - offset = 0xE10)

Bit	Symbol	Value	Description	Reset value
0	TXERR		Determines whether an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register.	
		0	No interrupt will be generated for a transmit error.	
		1	An interrupt will be generated when a transmit error occurs.	
1	RXERR		Determines whether an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register.	
		0	No interrupt will be generated for a receive error.	
		1	An interrupt will be generated when a receive error occurs.	

Table 680. FIFO interrupt enable set and read register (FIFOINTENSET - offset = 0xE10) *...continued*

Bit	Symbol	Value	Description	Reset value
2	TXLVL		Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	
		0	No interrupt will be generated based on the TX FIFO level.	
		1	If TXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.	
3	RXLVL		Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	
		0	No interrupt will be generated based on the RX FIFO level.	
		1	If RXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.	
31:4			Reserved. Read value is undefined, only zero should be written.	

35.6.13 FIFO interrupt enable clear and read register

The FIFOINTENCLR register is used to clear interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Table 681. FIFO interrupt enable clear and read (FIFOINTENCLR - offset = 0xE14)

Bit	Symbol	Description	Reset value
0	TXERR	Writing a one to this bit disables the TXERR interrupt.	0x0
1	RXERR	Writing a one to this bit disables the RXERR interrupt.	0x0
2	TXLVL	Writing a one to this bit disables the interrupt caused by the transmit FIFO reaching the level specified by the TXLVL field in the FIFOTRIG register.	0x0
3	RXLVL	Writing a one to this bit disables the interrupt caused by the receive FIFO reaching the level specified by the RXLVL field in the FIFOTRIG register.	0x0
31:4	-	Reserved. Read value is undefined, only zero should be written.	-

35.6.14 FIFO interrupt status register

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are both pending and currently enabled. This can simplify software handling of interrupts. Refer to the descriptions of interrupts in Section 35.6.9 “FIFO configuration register” and Section 35.6.10 “FIFO status register” for details.

Table 682. FIFO interrupt status register (FIFOINTSTAT - offset = 0xE18)

Bit	Symbol	Description	Reset value
0	TXERR	TX FIFO error.	0
1	RXERR	RX FIFO error.	0
2	TXLVL	Transmit FIFO level interrupt.	0
3	RXLVL	Receive FIFO level interrupt.	0
4	PERINT	Peripheral interrupt.	0
31:5	-	Reserved. Read value is undefined, only zero should be written.	-

35.6.15 FIFO write data register

The FIFOWR register is used to write values to be transmitted to the FIFO.

FIFOWR provides the possibility of altering some SPI controls at the same time as sending new data. For example, this can allow a series of SPI transactions involving multiple slaves to be stored in a DMA buffer and sent automatically.These added fields are described for bits 16 through 27 below.

Each FIFO entry holds data and associated control bits. Before data and control bits are pushed into the FIFO, the control bit settings can be modified. half-word writes to just the control bits (offset 0xE22) and does not push anything into the FIFO. A 0 written to the upper half-word will not modify the control settings. Non-zero writes to it will modify all the control bits. This is a write only register. Do not read-modify-write the register.

Byte, half-word or word writes to FIFOWR will push the data and control bits into the FIFO. Word writes with the upper half-word of 0, byte writes or half-word writes to FIFOWR will push the data and the current control bits, into the FIFO. Word writes with a non-zero upper half-word will modify the control bits before pushing them onto the stack.

To set-up a slave SPI for receive only, the control bit settings must be pushed into the write FIFO to become active. Therefore, at least one write to the FIFOWR data bits must be done to make the control bits active.

Table 683. FIFO write data register (FIFOWR - offset = 0xE20)

Bit	Symbol	Value	Description	Reset value
15:0	TXDATA		Transmit data to the FIFO.	-
16	TXSSEL0_N		Transmit slave select. This field asserts SSEL0 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL0 pin is configured by bits in the CFG register.	-
		0	SSEL0 asserted.	
		1	SSEL0 not asserted.	
17	TXSSEL1_N		Transmit slave select. This field asserts SSEL1 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL1 pin is configured by bits in the CFG register.	-
		0	SSEL1 asserted.	
		1	SSEL1 not asserted.	
18	TXSSEL2_N		Transmit slave select. This field asserts SSEL2 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL2 pin is configured by bits in the CFG register.	-
		0	SSEL2 asserted.	
		1	SSEL2 not asserted.	

Table 683. FIFO write data register (FIFOWR - offset = 0xE20) *…continued*

Bit	Symbol	Value	Description	Reset value
19	TXSSEL3_N		Transmit slave select. This field asserts SSEL3 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL3 pin is configured by bits in the CFG register.	-
		0	SSEL3 asserted.	
		1	SSEL3 not asserted.	
20	EOT		End of Transfer. The asserted SSEL will be de-asserted at the end of a transfer, and remain so for at least the time specified by the TRANSFER_DELAY value in the DLY register.	-
		0	SSEL not de-asserted. This piece of data is not treated as the end of a transfer. SSEL will not be de-asserted at the end of this data.	
		1	SSEL de-asserted. This piece of data is treated as the end of a transfer. SSEL will be de-asserted at the end of this piece of data.	
21	EOF		End of Frame. Between frames, a delay may be inserted, as defined by the FRAME_DELAY value in the DLY register. The end of a frame may not be particularly meaningful if the FRAME_DELAY value = 0. This control can be used as part of the support for frame lengths greater than 16 bits.	-
		0	Data not EOF. This piece of data transmitted is not treated as the end of a frame.	
		1	Data EOF. This piece of data is treated as the end of a frame, causing the FRAME_DELAY time to be inserted before subsequent data is transmitted.	
22	RXIGNORE		Receive Ignore. This allows data to be transmitted using the SPI without the need to read unneeded data from the receiver. Setting this bit simplifies the transmit process and can be used with the DMA.	-
		0	Read received data. Received data must be read first and then the RxData should be written to allow transmission to progress for non-DMA cases. SPI transmit will halt when the receive data FIFO is full. In slave mode, an overrun error will occur if received data is not read before new data is received.	
		1	Ignore received data. Received data is ignored, allowing transmission without the need to transmit data that is not needed.	

Table 683. FIFO write data register (FIFOWR - offset = 0xE20) *…continued*

Bit	Symbol	Value	Description	Reset value
23	TXIGNORE		Transmit Ignore. This allows data to be received using the SPI without the need to read unneeded data from the receiver. Setting this bit simplifies the transmit process and can be used with the DMA.This bit can only be set by writing to the upper 16 bits only of FIFOWR, i.e., a half-word write to offset 0xE22.	-
		0	Write transmit data. Transmit data must be written for each data exchange between master and slave. In slave mode, an underrun error occurs if transmit data is not provided before needed in a data frame.	
		1	Ignore transmit data. Data can be received without transmitting data (after FIFOWR has been initialized to set TXIGNORE). No transmitter flags are generated. When configured with TXIGNORE =1, the slave will set the data to always 0.	
27:24	LEN		Data Length. Specifies the data length from 4 to 16 bits. Note that transfer lengths greater than 16 bits are supported by implementing multiple sequential transmits. 0x0-2 = Reserved. 0x3 = Data transfer is 4 bits in length. 0x4 = Data transfer is 5 bits in length. ... 0xF = Data transfer is 16 bits in length.	-
31:28	-	-	Reserved. Read value is undefined, only zero should be written.	

35.6.16 FIFO read data register

The FIFORD register is used to read values that have been received by the FIFO.

Table 684. FIFO read data register (FIFORD - offset = 0xE30)

Bit	Symbol	Description	Reset value
15:0	RXDATA	Received data from the FIFO.	
16	RXSSEL0_N	Slave select for receive. This field allows the state of the SSEL0 pin to be saved along with received data. The value will reflect the SSEL0 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	
17	RXSSEL1_N	Slave select for receive. This field allows the state of the SSEL1 pin to be saved along with received data. The value will reflect the SSEL1 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	
18	RXSSEL2_N	Slave select for receive. This field allows the state of the SSEL2 pin to be saved along with received data. The value will reflect the SSEL2 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	
19	RXSSEL3_N	Slave select for receive. This field allows the state of the SSEL3 pin to be saved along with received data. The value will reflect the SSEL3 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	
20	SOT	Start of transfer flag. This flag will be 1 if this is the first data after the SSELs went from de-asserted to asserted (i.e., any previous transfer has ended). This information can be used to identify the first piece of data in cases where the transfer length is greater than 16 bits.	
31:21	-	Reserved. Read value is undefined, only zero should be written.	

35.6.17 FIFO data read with no FIFO pop register

This register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Table 685. FIFO data read with no FIFO pop (FIFORDNOPOP, offset = 0xE40)

Bit	Symbol	Description	Reset value
15:0	RXDATA	Received data from the FIFO.	-
16	RXSSEL0_N	Slave select for receive.	-
17	RXSSEL1_N	Slave select for receive.	-
18	RXSSEL2_N	Slave select for receive.	-
19	RXSSEL3_N	Slave select for receive.	-
20	SOT	Start of transfer flag.	-
31:21	-	Reserved. Read value is undefined, only zero should be written.	-

35.6.18 FIFO size register

The FIFOSIZE register provides the size FIFO for the selected Flexcomm function on this device.

Table 686. FIFO size register (FIFOSIZE - offset = 0xE48)

Bit	Symbol	Description	Reset value
4:0	FIFOSIZE	Provides the size of the FIFO for software. The size of the SPI FIFO is 8 entries.	0x08
31:5	-	Reserved.	-

35.6.19 Module identification register

The ID register identifies the type and revision of the SPI module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 687. Module identification register (ID, offset = 0xFFC)

Bit	Symbol	Description	Reset value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x0
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE020

35.7 Functional description

35.7.1 AHB bus access

With the exception of the FIFOWR register, the bus interface to the SPI registers contained in the Flexcomm Interface support only word writes. Byte and half-word writes are not supported in conjunction with the SPI function for those registers.

The FIFOWR register also supports byte and half-word (data only) writes in order to allow writing FIFO data without affecting the SPI control fields above bit 15. See Section 35.6.15 “FIFO write data register”

35.7.2 Operating modes: clock and phase selection

SPI interfaces typically allow configuration of clock phase and polarity. These are sometimes referred to as numbered SPI modes, as described in Table 688 and shown in Figure 121. CPOL and CPHA are configured by bits in the CFG register. See Section 35.6.2 “SPI configuration register”

Table 688. SPI mode summary

CPOL	CPHA	SPI Mode	Description	SCK rest state	SCK data change edge	SCK data sample edge
0	0	0	The SPI captures serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge.	low	falling	rising
0	1	1	The SPI changes serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge.	low	rising	falling
1	0	2	Same as mode 0 with SCK inverted.	high	rising	falling
1	1	3	Same as mode 1 with SCK inverted.	high	falling	rising

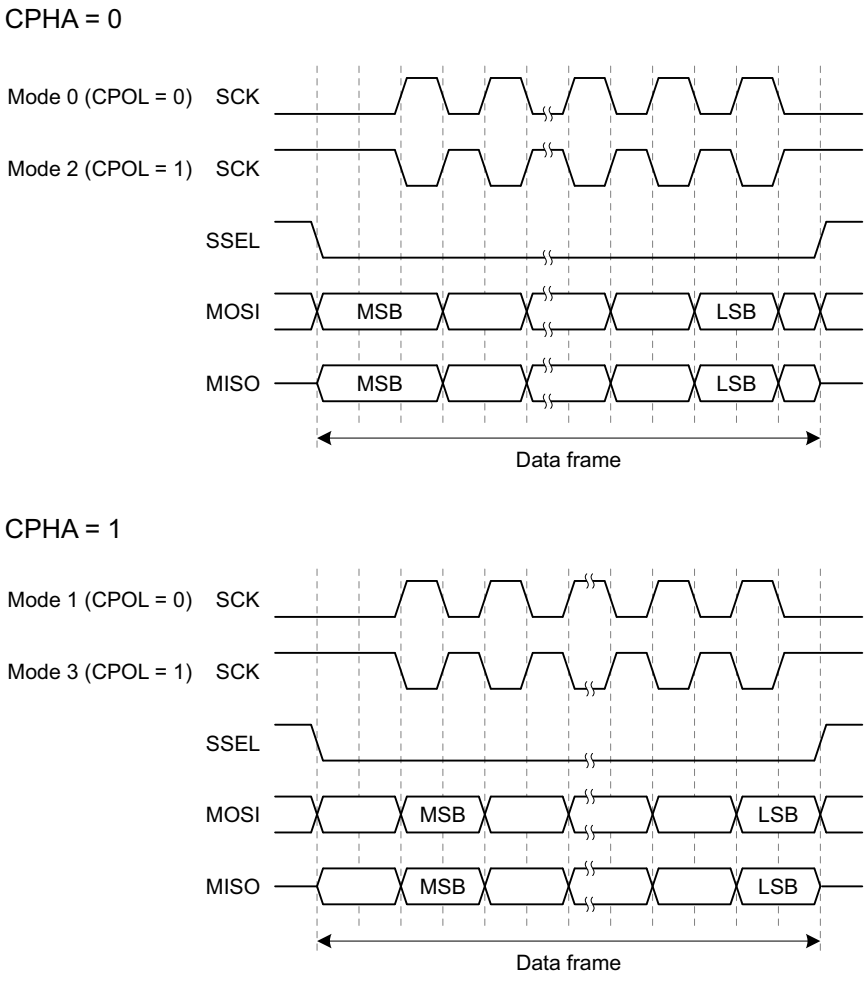


Fig 121. Basic SPI operating mode

35.7.3 Frame delays

Several delays can be specified for SPI frames. These include:

- Pre_delay: delay after SSEL is asserted before data clocking begins.
- Post_delay: delay at the end of a data frame before SSEL is de-asserted.
- Frame_delay: delay between data frames when SSEL is not de-asserted.
- Transfer_delay: minimum duration of SSEL in the de-asserted state between transfers.

35.7.3.1 Pre_delay and Post_delay

Pre_delay and Post_delay are illustrated by the examples in Figure 122. The Pre_delay value controls the amount of time between SSEL being asserted and the beginning of the subsequent data frame. The Post_delay value controls the amount of time between the end of a data frame and the de-assertion of SSEL..

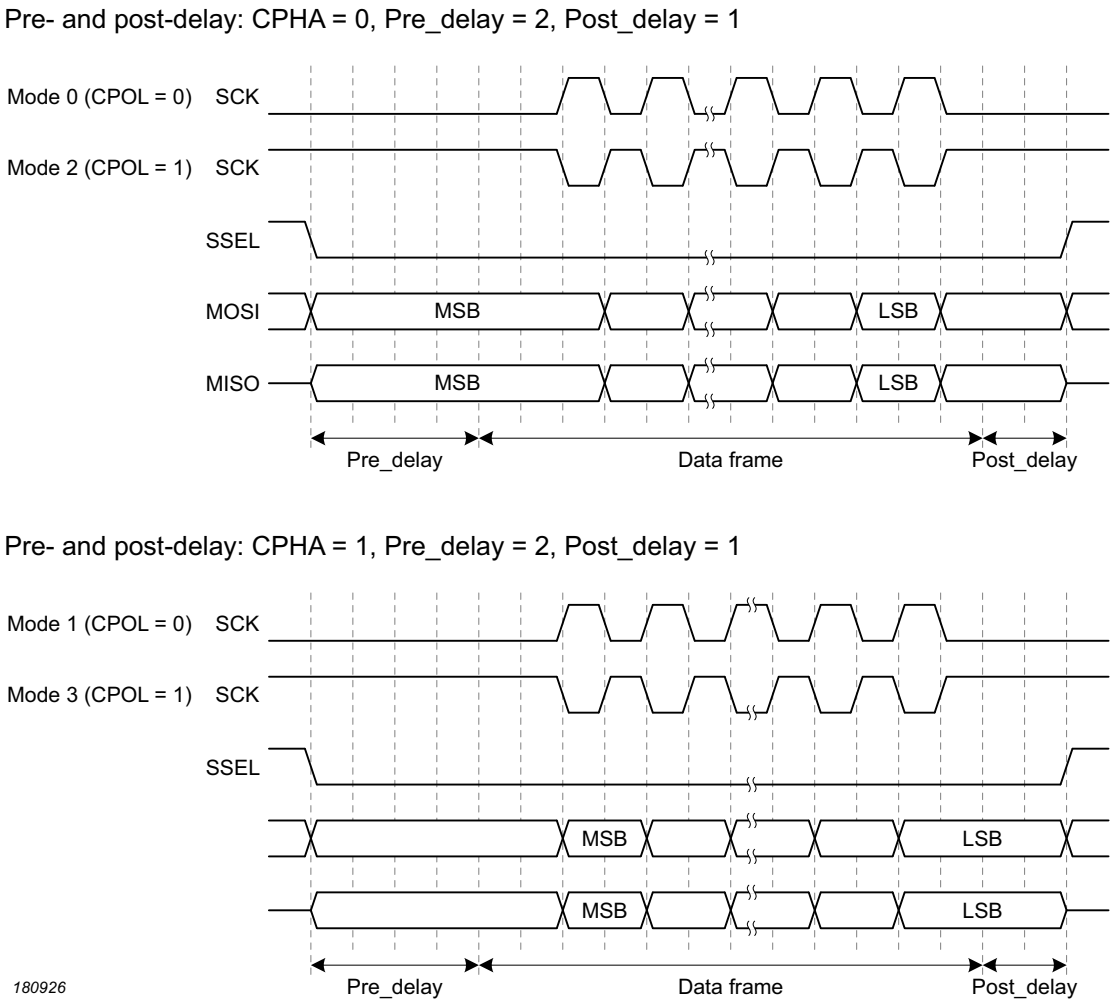


Fig 122. Pre_delay and Post_delay

35.7.3.2 Frame_delay

The Frame_delay value controls the amount of time at the end of each frame. This delay is inserted when the EOF bit = 1. Frame_delay is illustrated by the examples in Figure 123. Note that frame boundaries occur only where specified. This is because frame lengths can be of any size, involving multiple data writes. See Section 35.7.7 “Data lengths greater than 16 bits” for more information.

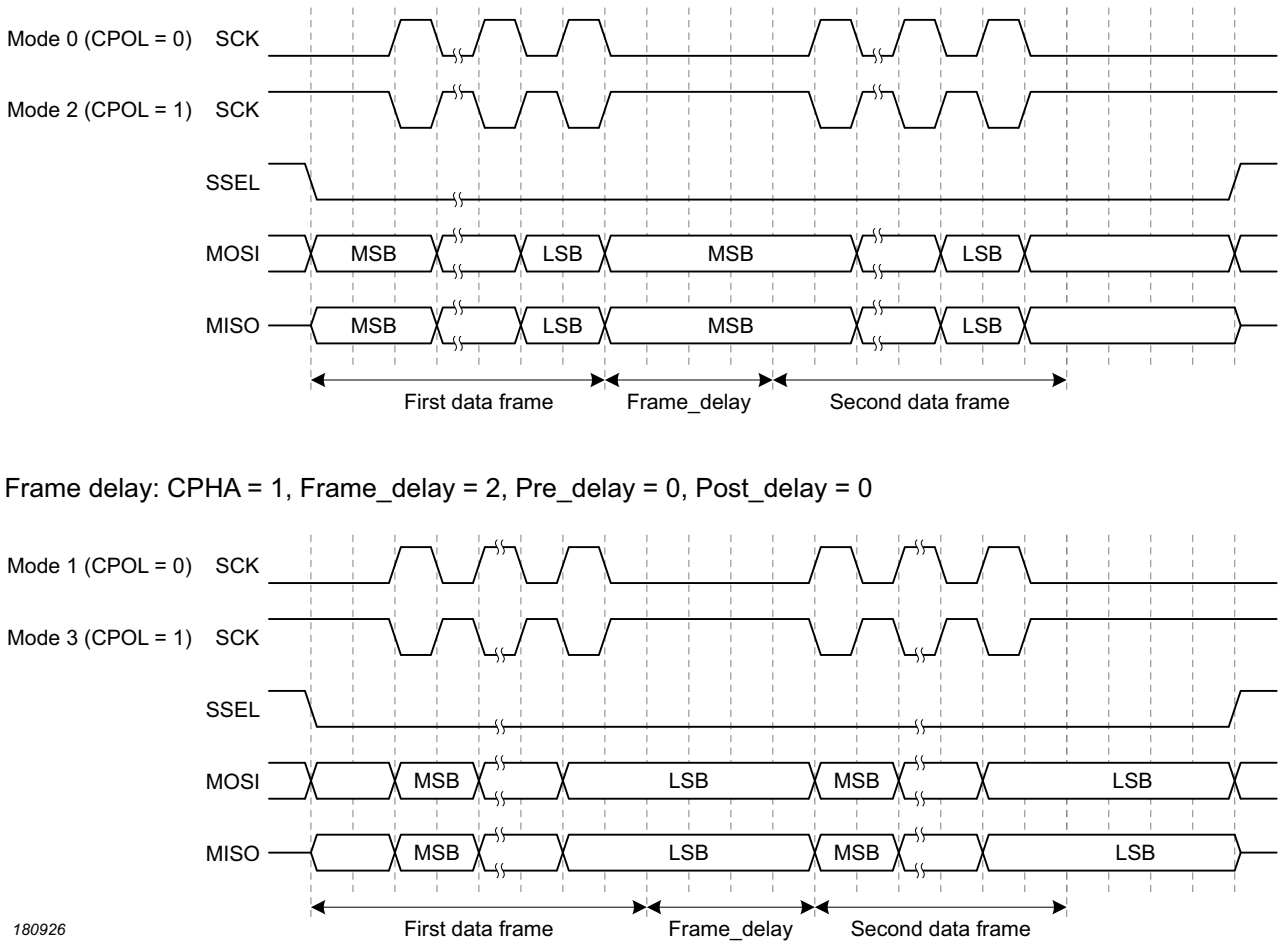


Fig 123. Frame_delay

35.7.3.3 Transfer_delay

The Transfer_delay value controls the minimum amount of time that SSEL is deasserted between transfers, because the EOT bit = 1. When Transfer_delay = 0, SSEL may be deasserted for a minimum of one SPI clock time. Transfer_delay is illustrated by the examples in Figure 124.

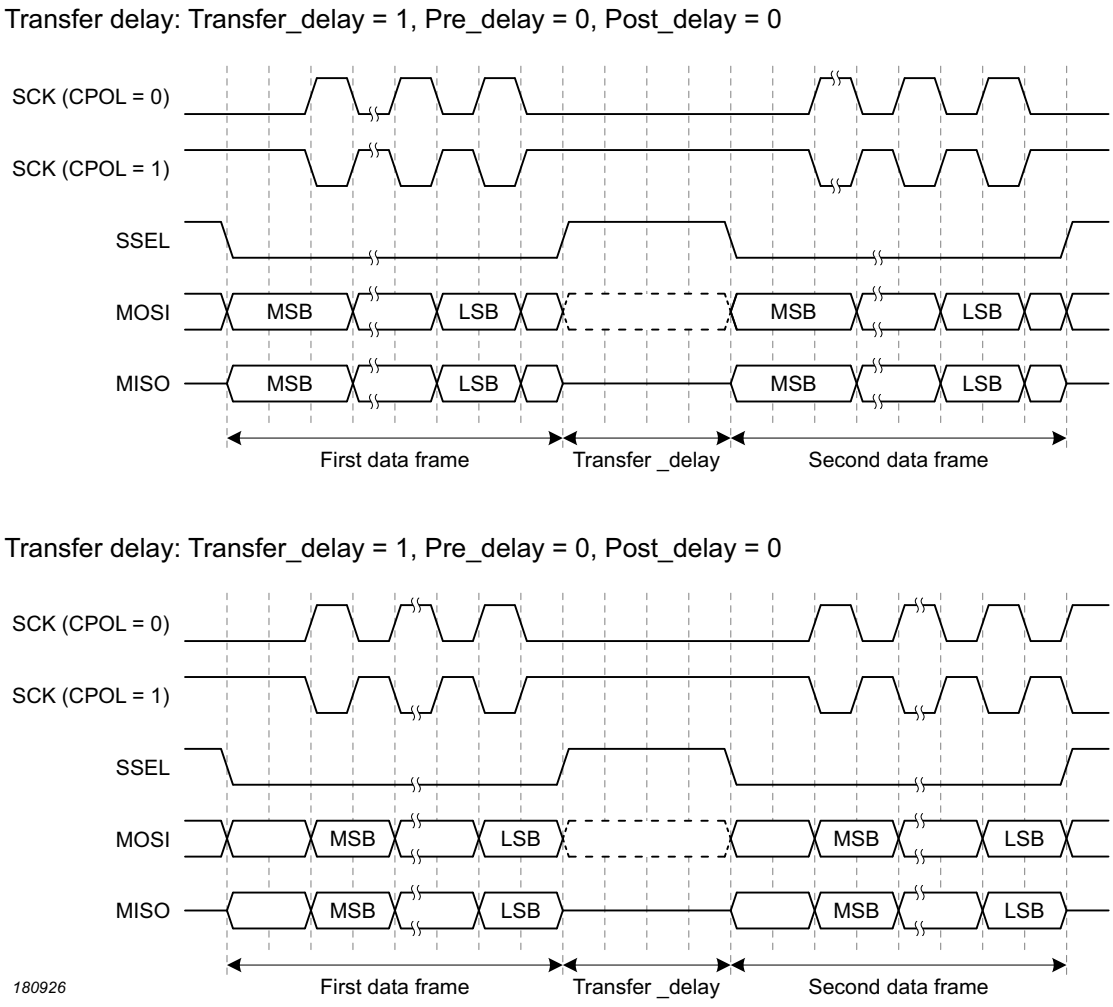


Fig 124. Transfer_delay

35.7.4 Clocking and date rates

In order to use the SPI, clocking details must be defined. This includes configuring the system clock and selection of the clock divider value in DIV. See Figure 22 “Clock generation for mass market devices <aaa-023922-mm get new number>”.

35.7.4.1 Data rate calculations

The SPI interface is designed to operate asynchronously from any on-chip clocks, and without the need for over-clocking.

In slave mode, this means that the SCK from the external master is used directly to run the transmit and receive shift registers and other logic.

In master mode, the SPI rate clock produced by the SPI clock divider is used directly as the outgoing SCK.

The SPI clock divider is an integer divider. The SPI in master mode can be set to run at the same speed as the selected FCLK, or at lower integer divide rates.

In slave mode, the clock is taken from the SCK input and the SPI clock divider is not used.

35.7.5 Slave select

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

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

In master mode, slave selects come from the TXSSEL bits in the FIFOWR register. In slave mode, the state of all four SSELs is saved along with received data in the RXSSEL_N field of the FIFORD register.

35.7.6 DMA operation

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

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

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

35.7.6.1 DMA master mode End-of-Transfer

When using polled or interrupt mode to transfer data in master mode, the transition to end-of-transfer status (drive SSEL inactive) is simple. The EOT bit of the FIFOWR control bits would be set just before or along with the writing of the last data to be sent.

When using the DMA in master mode, the End-of-Transfer status (drive SSEL inactive) can be generated in the following ways

1. Using DMA interrupt and a second DMA transfer

   To use only 8 or 16 bit wide DMA transfers for all the data, a second DMA transfer can be used to terminate the transfer (drive SSEL inactive).

   The transfer would be started by setting the control bits and then initiating the DMA transfer of all but the last byte/half word of data. The DMA completion interrupt function must modify the control bits to set EOT and then set-up DMA to send the last data.

2. Using DMA and SPI interrupts (or background SPI status polling)

   To use only one 8 or 16 bit wide DMA transfer for all the data, two interrupts would be required to properly terminate the transfer (drive SSEL inactive).

The SPI Tx DMA completion interrupt function sets the TXLVL field in the SPI FIFOTRIG register to 0 and sets the TXLVL interrupt enable bit in the FIFOINTENSET register.

The interrupt function handling the SPI TXLVL would set the SPI STAT register "END TRANSFER" bit, to force termination after all data output is complete.

3. Using DMA linked descriptor

   The DMA controller provides for a linked list of DMA transfer control descriptors. The initial descriptor(s) can be used to transfer all but the last data byte/half-word. These data transfers can be done as 8 or 16 bit wide DMA operations. A final DMA descriptor, linked to the first DMA descriptor, can be used to send the last data along with control bits to the FIFOWR register. The control bits would include the setting of the EOT bit.

   Note: The DMA interrupt function cannot set the SPI Status register (STAT) END TRANSFER control bit. This may terminate the transfer while the FIFO still has data to send.

4. Using 32 bit wide DMA

   Write both data and control bits to FIFOWR for all data. The control bits for the last entry would include the setting of the EOT bit. This also allows a series of SPI transactions involving multiple slaves with one DMA operation, by changing the TXSSELn_N bits.

35.7.7 Data lengths greater than 16 bits

The SPI interface handles data frame sizes from 4 to 16 bits directly. Larger sizes can be handled by splitting data up into groups of 16 bits or less. For example, 24 bits can be supported as two groups of 16 bits and 8 bits or two groups of 12 bits, among others. Frames of any size, including greater than 32 bits, can supported in the same way.

Details of how to handle larger data widths depend somewhat on other SPI configuration options. For instance, if it is intended for slave selects to be de-asserted between frames, then this must be suppressed when a larger frame is split into more than one part. Sending two groups of 12 bits with SSEL de-asserted between 24-bit increments, for instance, would require changing the value of the EOF bit on alternate 12-bit frames.

35.7.8 Data stalls

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

A stall for master receive can happen when a FIFO overflow (see RXERR in the FIFOSTAT register) would otherwise occur if the transmitter was not stalled. In modes 0 and 2, this occurs if the FIFO is full when the next piece of data is received. This stall happens one clock edge earlier than the transmitter stall.

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

Stalls are reflected in the STAT register by the stalled status flag, which indicates the current SPI status. The transmitter will be stalled until data is read from the receive FIFO. Use the RXIGNORE control bit setting to avoid the need to read the received data.

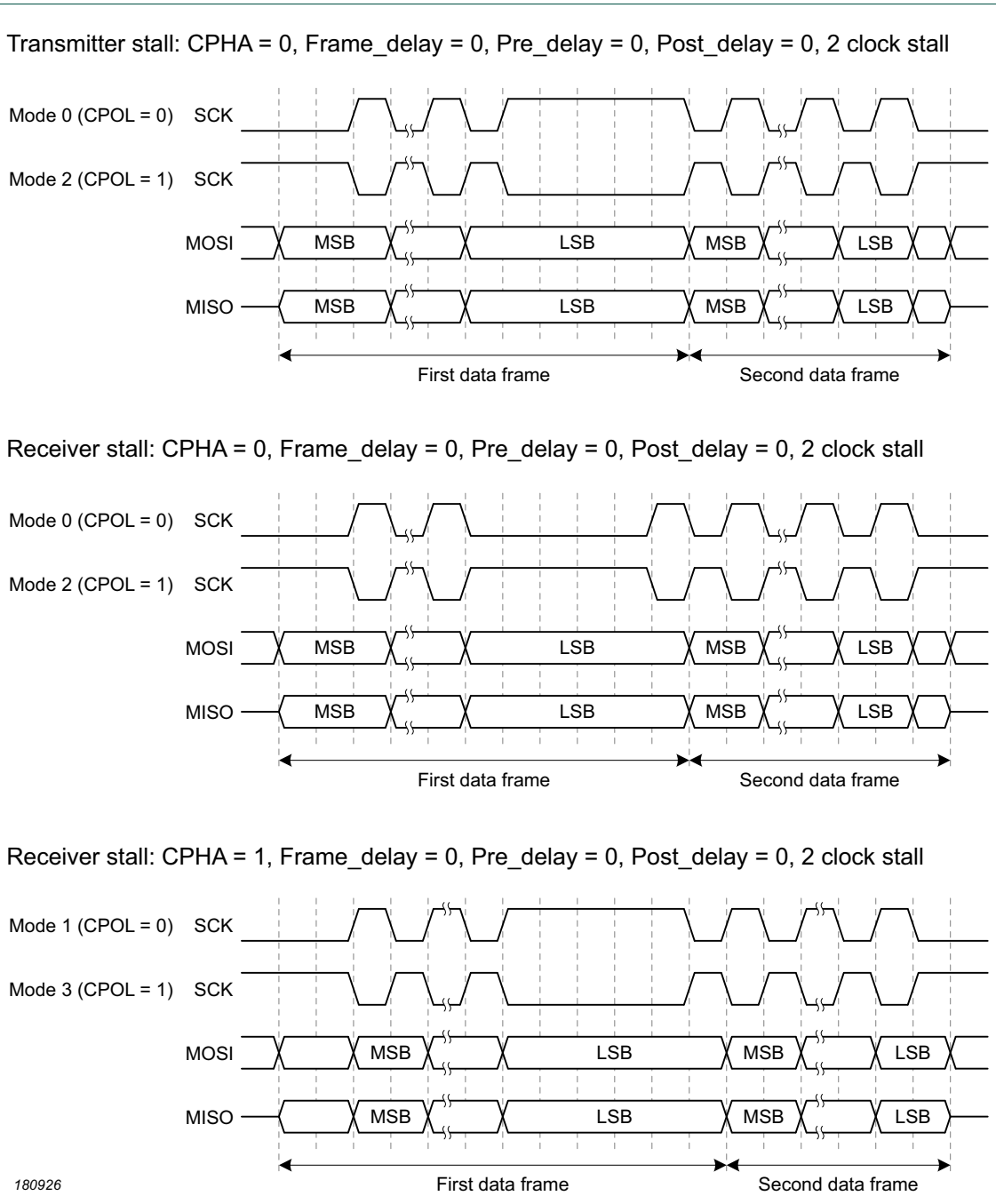


Fig 125. Examples of data stalls

36.1 How to read this chapter

The sys_ctrl contains I2S signal sharing. This feature is available on all LPC55S6x/LPC55S2x/LPC552x devices. A status register for USB HS is also present in sys_ctrl.

36.2 Features

- I2S signal sharing: allows multiple Flexcomm Interface I2S interfaces to share some combination of I2S clock, WS, and DATA without external board wiring.

36.3 Basic configuration

36.3.1 I2S signal sharing

Configure I2S signal sharing as follows.

Before writing FCnCTRLSEL and SHAREDCTRLSETx registers, remove write protection inside UPDATELCKOUT register by resetting bit UPDATELCKOUT.

1. Select the appropriate functions in IOCON for the pins that will actually be connected to the outside world for I2S operation.

Set up shared signal sets that will be used by writing to the SHAREDCTRLSET0 and/or SHAREDCTRLSET1 registers. See Section 36.5 “Register description”.

2. Set up any signal sharing for each Flexcomm Interface that uses shared signals by writing to the registers FC0CTRLSEL through FC7CTRLSEL as required.

Set up Flexcomm Interfaces using I2S signal sharing as needed, see Section 36.5 “Register description”. Any Flexcomm Interface acting as master first, then slaves.

Note: Signal sharing connections are made as register values are changed, without synchronization, and so should be done prior to the start of data streams.

Also, any I2S master that is providing SCK and WS signals for shared usage should also be configured to use the shared signal. For example, if Flexcomm Interface 0 is providing SCK and WS to shared set 0, FC0CTRLSEL should select shared set 0 for SCK and WS

36.4 Pin description

I2S signal sharing does not directly use pins, but offers additional internal routing of existing I2S pin functions.

36.5 Register description

Table 689. Register overview: sysctl (base address = 0x50023000)

Name	Access	Offset	Description	Reset value	Section
UPDATELCKOUT	RW	0x0	Update clock lock out.	undefined	36.5.1
FC0CTRLSEL	RW	0x40	Flexcomm Interface control selection N.	undefined	36.5.2
FC1CTRLSEL	RW	0x44	Flexcomm Interface control selection N.	undefined	36.5.2
FC2CTRLSEL	RW	0x48	Flexcomm Interface control selection N.	undefined	36.5.2
FC3CTRLSEL	RW	0x4C	Flexcomm Interface 3 is excluded from I2S sharing.	undefined	36.5.2
FC4CTRLSEL	RW	0x50	Flexcomm Interface control selection N.	undefined	36.5.2
FC5CTRLSEL	RW	0x54	Flexcomm Interface control selection N.	undefined	36.5.2
FC6CTRLSEL	RW	0x58	Flexcomm Interface control selection N.	undefined	36.5.2
FC7CTRLSEL	RW	0x5C	Flexcomm Interface control selection N.	undefined	36.5.2
SHAREDCTRLSET0	RW	0x80	Shared control set N.	undefined	36.5.3
SHAREDCTRLSET1	RW	0x84	Shared control set N.	undefined	36.5.3
USB_HS_STATUS	R	0x100	Peripheral enable register 0.	undefined	36.5.4

36.5.1 Update clock lock out register

This register is to prevent write access to all registers of sys_ctrl (except this one)

Table 690. Update clock lock out (UPDATELCKOUT, offset = 0x0)

Bit	Symbol	Value	Description	Reset value
0	UPDATELCKOUT		All registers.	0x0
		0	Normal mode; write enabled.	
		1	Protected mode; write disabled.	
31:1	-		Reserved.	undefined

36.5.2 Shared signal control select registers for each Flexcomm (0 to 7)

These registers select the SCK, WS, DATA input, and DATA output signal source for each Flexcomm Interface, excluding Flexcomm Interface 3. See Table 691 for details on how shared signals are connected and selected.

Table 691. Shared signal control select registers for each Flexcomm (FC0CTRLSEL to FC7CTRLSEL, offset 0x040 (FC0CTRLSEL) to 0x05C (FC7CTRLSEL))

Bit	Symbol	Value	Description	Reset value
1:0	SCKINSEL		Selects the source for SCK going into this Flexcomm.	0x0
		0	Selects the dedicated FCn_SCK function for this Flexcomm.	
		1	SCK is taken from shared signal set 0 (defined by SHAREDCTRLSET0).	
		2	SCK is taken from shared signal set 1 (defined by SHAREDCTRLSET1).	
		3	Reserved.	
7:2	-		Reserved.	undefined

Table 691. Shared signal control select registers for each Flexcomm (FC0CTRLSEL to FC7CTRLSEL, offset 0x040 (FC0CTRLSEL) to 0x05C (FC7CTRLSEL)) *…continued*

Bit	Symbol	Value	Description	Reset value
9:8	WSINSEL		Selects the source for WS going into this Flexcomm.	0x0
		0	Selects the dedicated (FCn_TXD_SCL_MISO_WS) function for this Flexcomm.	
		1	WS is taken from shared signal set 0 (defined by SHAREDCTRLSET0).	
		2	WS is taken from shared signal set 1 (defined by SHAREDCTRLSET1).	
		3	Reserved.	
15:10	-		Reserved.	
17:16	DATAINSEL		Selects the source for DATA input to this Flexcomm.	0x0
		0	Selects the dedicated FCn_RXD_SDA_MOSI_DATA input for this Flexcomm.	
		1	Input data is taken from shared signal set 0 (defined by SHAREDCTRLSET0).	
		2	Input data is taken from shared signal set 1 (defined by SHAREDCTRLSET1).	
		3	Reserved.	
23:18	-		Reserved.	undefined
25:24	DATAOUTSEL		Selects the source for DATA output from this Flexcomm.	0x0
		0	Selects the dedicated FCn_RXD_SDA_MOSI_DATA output from this Flexcomm.	
		1	Output data is taken from shared signal set 0 (defined by SHAREDCTRLSET0).	
		2	Output data is taken from shared signal set 1 (defined by SHAREDCTRLSET1).	
		3	Reserved.	
31:26	-		Reserved.	undefined

36.5.3 Control registers for each set of shared signals

These registers select the sources of SCK, WS, and DATA input for the two shared signal groups, and selects which Flexcomm Interfaces participate in shared DATA outputs. See and for details of how shared signals are connected and selected.

Table 692. Shared control set N (SHAREDCTRLSET0, offset = 0x80) and (SHAREDCTRLSET1, offset = 0x84)

Bit	Symbol	Value	Description	Reset value
2: 0	SHAREDSCKSEL		Selects the source for SCK of this shared signal set.	0x0
		0	SCK for this shared signal set comes from Flexcomm 0.	
		1	SCK for this shared signal set comes from Flexcomm 1.	
		2	SCK for this shared signal set comes from Flexcomm 2.	
		3	Reserved.	
		4	SCK for this shared signal set comes from Flexcomm 4.	
		5	SCK for this shared signal set comes from Flexcomm 5.	
		6	SCK for this shared signal set comes from Flexcomm 6.	
		7	SCK for this shared signal set comes from Flexcomm 7.	
3			Reserved	undefined
6: 4	SHAREDWSSEL		Selects the source for WS of this shared signal set.	0x0
		0	WS for this shared signal set comes from Flexcomm 0.	
		1	WS for this shared signal set comes from Flexcomm 1.	
		2	WS for this shared signal set comes from Flexcomm 2.	
		3	Reserved.	
		4	WS for this shared signal set comes from Flexcomm 4.	
		5	WS for this shared signal set comes from Flexcomm 5.	
		6	WS for this shared signal set comes from Flexcomm 6.	
		7	WS for this shared signal set comes from Flexcomm 7.	
7			Reserved	undefined
10: 8	SHAREDDATASEL		Selects the source for DATA input for this shared signal set.	0x0
		0	DATA input for this shared signal set comes from Flexcomm 0.	
		1	DATA input for this shared signal set comes from Flexcomm 1.	
		2	DATA input for this shared signal set comes from Flexcomm 2.	
		3	Reserved.	
		4	DATA input for this shared signal set comes from Flexcomm 4.	
		5	DATA input for this shared signal set comes from Flexcomm 5.	
		6	DATA input for this shared signal set comes from Flexcomm 6.	
		7	DATA input for this shared signal set comes from Flexcomm 7.	
15:11			Reserved	undefined
16	FC0DATAOUTEN		Controls FC0 contribution to SHAREDDATAOUT for this shared set.	0x0
		0	Data output from FC0 does not contribute to this shared set.	
		1	Data output from FC0 does contribute to this shared set.	
17	FC1DATAOUTEN		Controls FC1 contribution to SHAREDDATAOUT for this shared set.	0x0
		0	Data output from FC1 does not contribute to this shared set.	
		1	Data output from FC1 does contribute to this shared set.	
18	FC2DATAOUTEN		Controls FC2 contribution to SHAREDDATAOUT for this shared set.	0x0
		0	Data output from FC2 does not contribute to this shared set.	
		1	Data output from FC2 does contribute to this shared set.	
19			Reserved	undefined

Table 692. Shared control set N (SHAREDCTRLSET0, offset = 0x80) and (SHAREDCTRLSET1, offset = 0x84)

Bit	Symbol	Value	Description	Reset value
20	FC4DATAOUTEN		Controls FC4 contribution to SHAREDDATAOUT for this shared set.	0x0
		0	Data output from FC4 does not contribute to this shared set.	
		1	Data output from FC4 does contribute to this shared set.	
21	FC5DATAOUTEN		Controls FC5 contribution to SHAREDDATAOUT for this shared set.	0x0
		0	Data output from FC5 does not contribute to this shared set.	
		1	Data output from FC5 does contribute to this shared set.	
22	FC6DATAOUTEN		Controls FC6 contribution to SHAREDDATAOUT for this shared set.	0x0
		0	Data output from FC6 does not contribute to this shared set.	
		1	Data output from FC6 does contribute to this shared set.	
23	FC7DATAOUTEN		Controls FC7 contribution to SHAREDDATAOUT for this shared set.	0x0
		0	Data output from FC7 does not contribute to this shared set.	
		1	Data output from FC7 does contribute to this shared set.	
31:24			Reserved	undefined

36.5.4 Status register for USB HS

This register shows low voltage detection signal for USB HS 3.3V supply domain. Power status detector for this domain is internal to USB PHY.

Table 693. Flexcomm Interface control selection N (FC2CTRLSEL, offset = 0x48)

Bit	Symbol	Value	Description	Reset value
0	USBHS_3V_NOK		USB_HS: Low voltage detection on 3.3V supply. Active low isolation.	undefined
		0	3.3Vsupply is good.	
		1	3.3V supply is too low.	
31:01			Reserved.	undefined

36.6 Functional description

36.6.1 I²S signal sharing

The I²S signal sharing features are available on all LPC55S6x/LPC55S2x/LPC552x devices.

It is sometimes desirable to use multiple I²S functions together in a single TDM stream without sacrificing more pins than are needed. I²S signal sharing allows this kind of use without the need for external connections to multiple pins outside of the device. Note that this is only needed when the requirements exceed what can be accomplished with a single I²S interface that includes four channel pairs.

Signal sharing allows more than one on-chip I²S interface to be connected to clock, WS, and input data on the same pins without external board wiring. Multiple I²S functions contributing output data to a single data line must still be accomplished with an external connection.

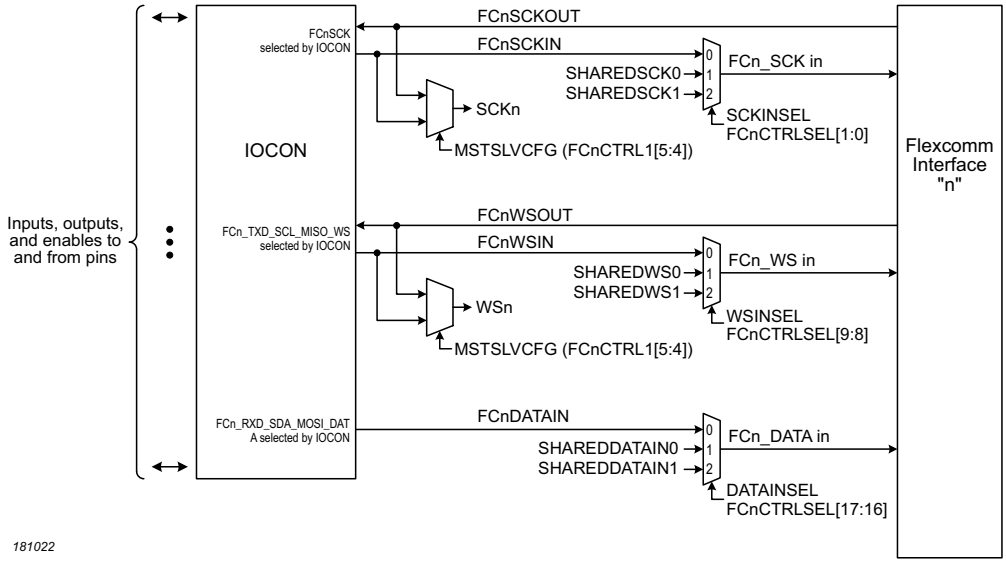
In general, each Flexcomm Interface configured for I²S can choose:

- Its own SCK, or a shared SCK.
- Its own WS, or a shared WS.
- Its own DATA in, or a shared DATA in.

Each Flexcomm Interface potentially contributes to shared signals:

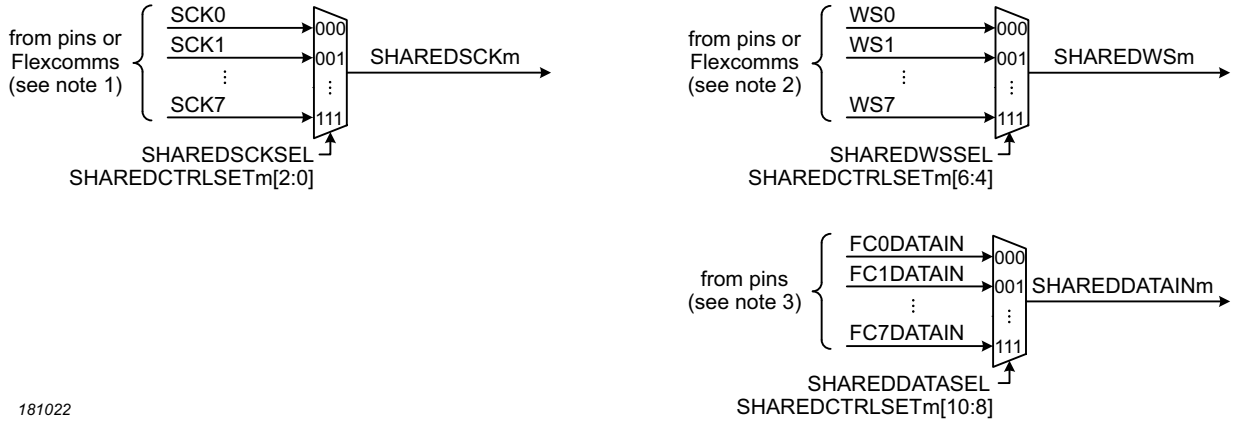
- Its own SCK (in or out, depending on whether it is a master or slave).
- Its own WS (in or out, depending on whether it is a master or slave).
- Its own DATA input.

Representative logic for the connection possibilities are shown in Figure 126 and Figure 127.



Note: these connection options are replicated for each Flexcomm "n", for all Flexcomm Interfaces with I2S support.

Fig 126. Shared signal connections for each Flexcomm Interface



Notes: This logic is replicated for each of the two sets of shared signals ("m", 0 through 1).

Each SCK or WS comes either from IOCON (whatever pin is selected as FCn_SCK) or from the related Flexcomm Interface if the Flexcomm is configured as an I2S master.

Each WS comes either from IOCON (whatever pin is selected as FCn_TXD_SCL_MISO_WS) or from the related Flexcomm Interface if the Flexcomm is configured as an I2S master.

Each FCnDATAIN comes from IOCON (whatever pin is selected as FCn_RXD_SDA_MOSI_DATA)

Fig 127. Shared signal source selection and control

36.6.1.1 Examples

Figure 128 shows a simple example of a bidirectional codec with input and output data connected to two different I^{2}S interfaces, using signal sharing to reduce connections to a single SCK and single WS pin. In this case, one I^{2}S interface is a master transmitter and one is a slave receiver. Data input ands output cannot be shared on one pin because they

are separate pins on the external codec

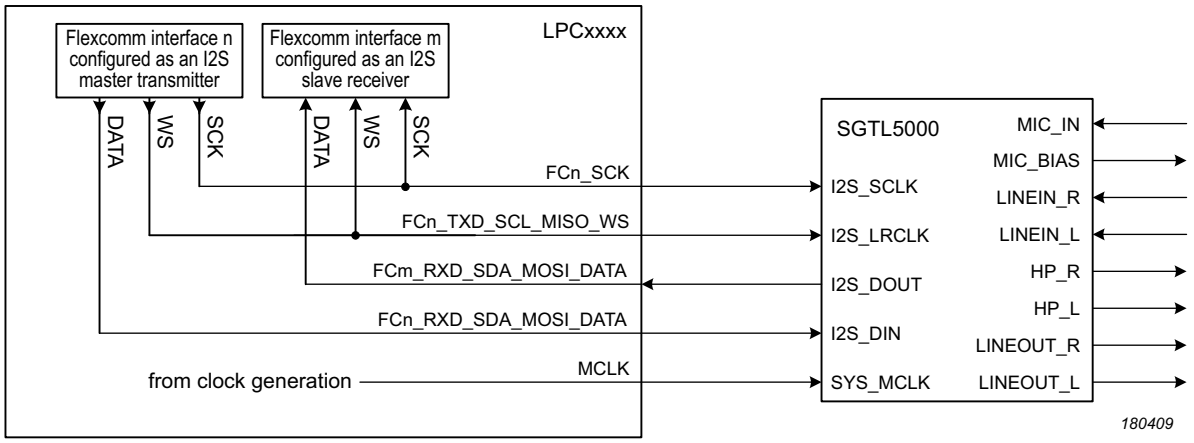
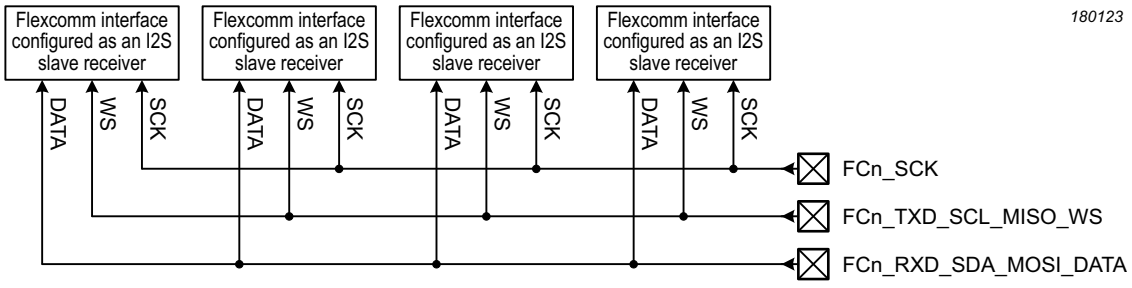


Fig 128. Example connection to an I²S bidirectional codec

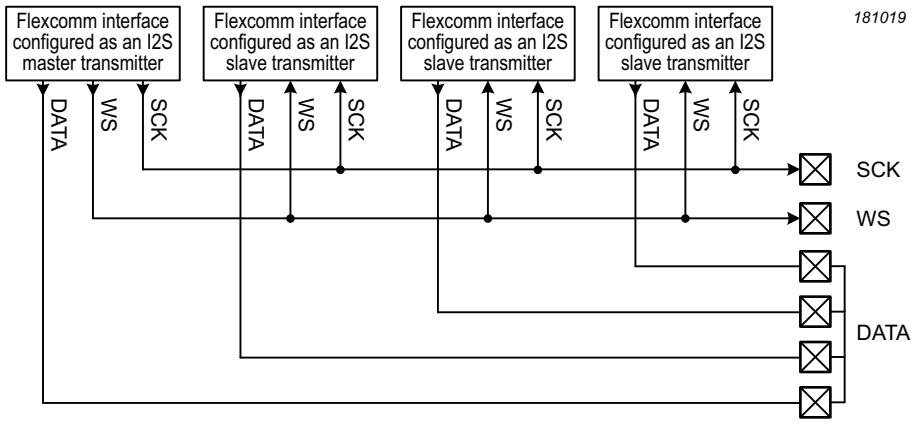
Figure 129 shows a generic case of multiple slaves and/or receivers sharing SCK and WS, and/or DATA. This scenario includes received data sharing (e.g. different I²S interfaces receiving data from different slots in a TDM stream).



All I2S interfaces are slave receivers sharing SCK, WS, and DATA.

Fig 129. I²S signal sharing example, multiple slave receivers

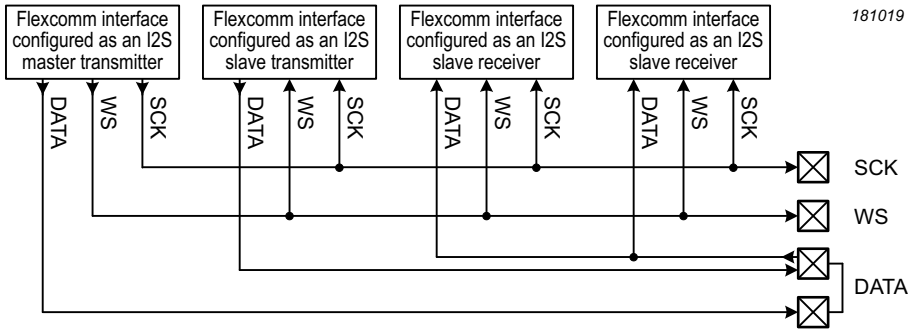
Figure 130 shows master to slave operation where one I²S interface is a master going off chip, and other on-chip I²S interfaces are slaved to it. Data could be either transmitted or received. Multiple I²S interfaces supply data to a single stream by wiring multiple pins together.



All I2S interfaces are transmitters sharing SCK and WS. One I2S interfaces is the master, others are slaves. Data is to/from multiple I2S interfaces. Here the I2S interfaces are shown as transmitters, but could be receivers as in example 1.

Fig 130. I²S signal sharing example, one master and multiple slave transmitters

Figure 131 shows data with one I²S interface transmitting onto a shared DATA line while at least one other I²S is receiving from the same DATA line. This does not necessarily mean that the transmitted data is what is being received. They could be different packets in a TDM frame. The example shows two I²S interfaces transmitting and two receiving, but it could be any combination.



All I2S interfaces share SCK and WS. One I2S interfaces is the master, others are slaves. Two I2S interfaces are transmitters, others are receivers. Data is output from two I2S interfaces and input to two I2S interfaces. Which I2S interfaces are transmitters and receivers is arbitrary in this example.

Fig 131. I²S signal sharing example, one master with mixed transmitters and receivers

37.1 How to read this chapter

I2S functionality is available on all LPC55S6x/LPC55S2x/LPC552x devices. I2S is a function that is implemented in selected Flexcomm Interfaces. The I2S function will be enabled in all Flexcomm Interfaces, each with 1x I2S channel pair (each I2S channel pair can handle transmitted or received stereo data).

The I2S channel pairs in a single Flexcomm Interface share 1 SCK, 1 data line, 1 WS. MCLK in or out is handled outside of the Flexcomm Interface. See clocking diagram for I2S clocking.

I2S pin sharing logic to support full-duplex I2S operation from common pins is described in Chapter 36 “LPC55S6x/LPC55S2x/LPC552x Sys_ctrl”.

37.2 Features

The I2S bus provides a standard communication interface for streaming data transfer applications such as digital audio or data collection. The I2S bus specification defines a 3-wire serial bus, having one data, one clock, and one word select/frame trigger signal, providing single or dual (mono or stereo) audio data transfer as well as other configurations.

The I2S interface within one Flexcomm Interface provides at least one channel pair that can be configured as a master or a slave. Other channel pairs, if present, always operate as slaves. All of the channel pairs within one Flexcomm Interface share one set of I2S signals, and are configured together for either transmit or receive operation, using the same mode, same data configuration and frame configuration. All such channel pairs can participate in a time division multiplexing (TDM) arrangement. For cases requiring an MCLK input and/or output, it is handled outside of the I2S block in the system level clocking scheme.

- A Flexcomm Interface may implement one or more I2S channel pairs. The channel pair can be either a master or a slave, and the rest of the channel pairs are always slaves.
- Configurable data size for all channels within one Flexcomm Interface, from 4 bits to 32 bits. Each channel pair can also be configured independently to act as a single channel (mono as opposed to stereo operation).
- A channel pair within one Flexcomm Interface share a single bit clock (SCK) and word select/frame trigger (WS), and data line (SDA).
- Data for all I2S traffic within one Flexcomm Interface uses the Flexcomm Interface FIFO. The FIFO depth is eight entries.
- Left justified and right justified data modes.
- DMA support using FIFO level triggering.
- TDM (Time Division Multiplexing) with a several stereo slots and/or mono slots is supported. Each channel pair can act as any data slot. Multiple channel pairs can participate as different slots on one TDM data line.

- The bit clock and WS can be selectively inverted.
- Sampling frequencies supported, depends on the specific device configuration and applications constraints for example, system clock frequency and PLL availability, but generally supports standard audio data rates.

37.3 Basic configuration

Initial configuration of the I2S peripheral is accomplished as follows:

1. Peripheral clock: Make sure that the related Flexcomm Interface is enabled in the AHBCLKCTRL1 register. See Section 4.5.18 “AHB clock control 1”.
2. Flexcomm Interface clock: Select a clock source for the related Flexcomm Interface. Options are shown in Figure 6. Also see Section 4.5.41 “Flexcomm Interface clock source select registers”.

   Remark: The Flexcomm Interface function clock frequency **(at the output of the FRG associated with the flexcomm)** shall not be above 100 MHz.
3. If required, use the PRESETCTRL1 register, see Table 46 to reset the Flexcomm Interface that is about to have a specific peripheral function selected.
4. Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface. See Section 32.7.1 “Peripheral Select and Flexcomm Interface ID register”.
5. Pins: Make sure that the IOCON block is enabled in the AHBCLKCTRL0 register, see Section 4.5.17 “AHB clock control 0”. Select I2S pins and pin modes through the relevant IOCON registers, see Chapter 15 “LPC55S6x/LPC55S2x/LPC552x I/O Pin Configuration (IOCON)”.
6. I2S rate: For master operation, the I2S rate is determined by the clock selected in step 2 above, optionally modified using the DIV register, see Table 700. Slave functions typically use the incoming I2S clock directly.
7. Interrupts: To enable I2S channel pair interrupts, see FIFOINENSET in Section 37.7.8 “FIFO interrupt enable set and read”, FIFOINTENCLR in Section 37.7.9 “FIFO interrupt enable clear and read”, and FIFOINTSTAT in Section 37.7.10 “FIFO interrupt status register”. The related Flexcomm Interface interrupt must be enabled in the NVIC using the appropriate interrupt set enable register, see Chapter 7 “LPC5500 Nested Vectored Interrupt Controller (NVIC)”.
8. DMA: I2S channel pair master and slave functions can operated with the system DMA controller, see Chapter 16 “LPC5500 DMA controller”, and must be enabled in the FIFOCFG register, see Section 37.7.5 “FIFO configuration register”.

37.4 Architecture

See Figure 132 for the overall architecture of an example I2S subsystem.

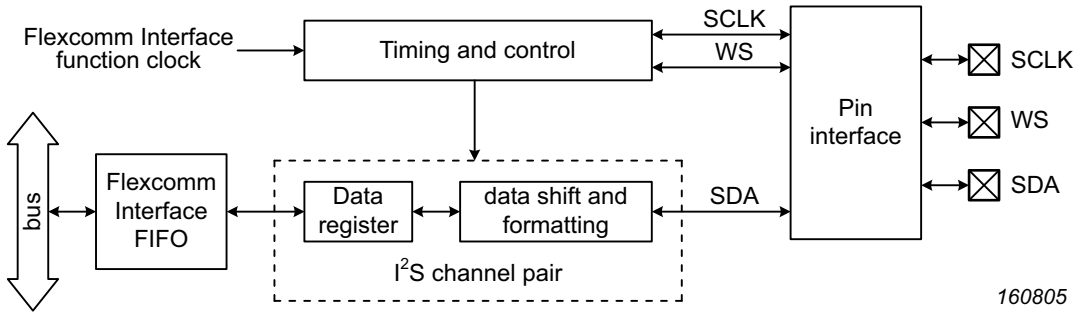


Fig 132. I2S block diagram

37.5 Terminology

Table 694. List of the terminologies used in the document

Term	Description
Channel	One piece of information on a single SDA line. In classic I2S, there is a single set of stereo data, which are two channels (left and right). In TDM modes, there may be many channels on a single SDA line.
Channel Pair	Two channels of data can be carried on one wire in classic I2S: left and right. On a microcontroller, it is typically what is implemented in a single instance of an I2S interface.
Classic I2S	The term used in this document, is in reference to the original I2S bus specification from Philips Semiconductors. The specification defines two channel stereo data on SDA, where the WS state identifies the left (low) and right (high) channel, and data is delayed by one clock after WS transitions. The many variations of I2S that may be found have descended from its original specification.
DSP mode	DSP mode packs channel data together in the bit stream (left data followed by right data for each slot) and does not use WS to identify left and right data. WS may be a single SCK pulse, or a single data slot long pulse, in addition to a 50% duty cycle pulse. It may be used in conjunction with TDM mode.
MCLK	Master clock. In some I2S systems, it is provided as a multiple of the sample rate (fs), higher than the bit rate, such as 256 fs. Devices could potentially use this clock to construct a bit clock, or for internal operations such as data filtering.
SCK	Serial clock. Sometimes referred to as BCK. It is a bit clock for data on the SDA line.
SDA	Serial data. A single SDA provides one data stream, which may have many formats.
Slot	One data position in an I2S stream, typically each with the same slot length. For classic I2S, there is only one slot for stereo data. In a TDM mode, there can be several slots. In MONO mode, each slot is defined as one piece of data, rather than both left and right data.
TDM mode	TDM mode uses multiple data slots in order to put more channels of data into a single stream. It may be used in conjunction with DSP mode or I2S mode.
WS	Word select. Sometimes called LRCLK, distinguishes left versus right data in most single stereo formats. It is used as a frame delimiter in DSP and TDM modes.

37.6 Pin description

Remark: When the I2S function is outputting SCK and/or WS, it uses a return signal from the related pin to adjust internal timing. In order for the I2S to operate, the signals must be connected to a device pin, via IOCON selection.

Table 695. I2S pin description

Pin	Type	Name used in pin configuration chapter	Description
SCK	I/O	FCn_SCK	Serial clock for I2Sn. Clock signal used to synchronize the transfer of data on the SDA pin. It is driven by the master and received by one or more slaves. **Remark:** When the primary I2S channel pair of a Flexcomm Interface is configured as a master, so that SCK is an output, it must be connected to a pin for the I2S to work properly.
WS	I/O	FCn_TXD_SCL_MISO	Word select for I2Sn. Synchronizing signal for the beginning of each data frame and, in some modes, left vs right channel data. It is driven by the master and received by one or more slaves. **Remark:** When the primary I2S channel pair of a Flexcomm Interface is configured as a master, so that WS is an output, it must be connected to a pin for the I2S to work properly.
SDA	I/O	FCn_RXD_SDA_MOSI	Serial data for a single data stream used by one or more I2S channel pairs of I2Sn. The format of data is configurable. It is driven by one or more transmitters and read by one or more receivers.
MCLK	I/O	MCLK	Master clock. A multiple of the sample clock can optionally be provided by a master to other devices in the system, or can be received and divided down within a Flexcomm Interface to locally generate SCK and/or WS. This clock is not created inside the I2S block. If MCLK is supported as an input to the device, it can be routed to the I2S block and used to operate its functions. If MCLK is an output from the device, the clock that is used to create that MCLK can also be routed to the I2S block and used to operate its functions.

37.7 Register description

The registers shown in Table 696 apply if the I²S function is selected in a Flexcomm Interface that supports I²S. The primary channel pair uses registers as shown under the row heading *Registers for the primary channel pair and shared registers*, followed by FIFO related registers. Registers for any additional channel pairs are shown under the row heading *Registers for secondary channel pairs:*

The reset value reflects the value of defined bits only, and does not include reserved bits:

I²S0 base address: 4008_6000h

I²S1 base address: 4008_7000h

I²S2 base address: 4008_8000h

I²S3 base address: 4008_9000h

I²S4 base address: 4008_A000h

I²S5 base address: 4009_6000h

I²S6 base address: 4009_7000h

I²S7 base address: 4009_8000h

Table 696. Register overview for the I²S function of one Flexcomm Interface

Name	Access	Offset [1]	Description	Reset value	Section
Registers for the data channel pair and shared registers					
CFG1	R/W	0xC00	Configuration register 1 for the primary channel pair.	0	37.7.1
CFG2	R/W	0xC04	Configuration register 2 for the primary channel pair.	0	37.7.2
STAT	RO/W1C	0xC08	Status register for the primary channel pair.	0	37.7.3
DIV	R/W	0xC1C	Clock divider, used by all channel pairs.	0	37.7.4
Registers for FIFO control and data access					
FIFOCFG	R/W	0xE00	FIFO configuration and enable.	0x0E00	37.7.5
FIFOSTAT	R/W	0xE04	FIFO status.	0x18	37.7.6
FIFOTRIG	R/W	0xE08	FIFO trigger settings for interrupt and DMA request.	0	37.7.7
FIFOINTENSET	R/W1C	0xE10	FIFO interrupt enable set (enable) and read.	0	37.7.8
FIFOINTENCLR	R/W1C	0xE14	FIFO interrupt enable clear (disable) and read.	0	37.7.9
FIFOINTSTAT	RO	0xE18	FIFO interrupt status.	0	37.7.10
FIFOWR	WO	0xE20	FIFO write data.	-	37.7.11
FIFOWR48H	WO	0xE24	FIFO write data for upper data bits. It may only be used if the I²S is configured for 2x 24-bit data and not using DMA.	-	37.7.12
FIFORD	RO	0xE30	FIFO read data.	-	37.7.13
FIFORD48H	RO	0xE34	FIFO read data for upper data bits. It may only be used if the I²S is configured for 2x 24-bit data and not using DMA.	-	37.7.14
FIFORDNOPOP	RO	0xE40	FIFO data read with no FIFO pop.	-	37.7.15

Table 696. Register overview for the I2S function of one Flexcomm Interface *…continued*

Name	Access	Offset [1]	Description	Reset value	Section
FIFORD48HNOPOP	RO	0xE44	FIFO data read for upper data bits with no FIFO pop. It may only be used if the I2S is configured for 2x 24-bit data and not using DMA.	-	37.7.16
FIFOSIZE	R	0xE48	FIFO size.	0x8	
ID register:					
ID	RO	0xFFC	I2S module identification. This value appears in the shared Flexcomm Interface peripheral ID register when I2S is the selected function.	0xE090	37.7.18

[1] Offset is within the related Flexcomm Interface address space.

37.7.1 Configuration register 1

The CFG1 register contains mode settings, most of which apply to all I2S channel pairs within one Flexcomm Interface. A few settings apply only to the primary channel pair, as noted.

Table 697. Configuration register 1 (CFG1, offset = 0xC00)

Bit	Symbol	Value	Description	Reset value
0	MAINENABLE		Main enable for I2S function in this Flexcomm Interface	0
		0	All I2S channel pairs in this Flexcomm Interface are disabled and the internal state machines, counters, and flags are reset. No other channel pairs can be enabled.	
		1	This I2S channel pair is enabled. Other channel pairs in this Flexcomm Interface may be enabled in their individual PAIRENABLE bits.	
1	DATAPAUSE		Data flow pause. Allows pausing data flow between the I2S serializer/deserializer and the FIFO. It can be done in order to change streams, or while restarting after a data underflow or overflow. When paused, FIFO operations can be done without corrupting data that is in the process of being sent or received. Once a data pause has been requested, the interface may need to complete sending data that was in progress before interrupting the flow of data. Software must check that the pause is actually in effect before taking action. It is done by monitoring the DATAPAUSED flag in the STAT register. When DATAPAUSE is cleared, data transfer will resume at the beginning of the next frame.	0
		0	Normal operation, or resuming normal operation at the next frame if the I2S has already been paused.	
		1	A pause in the data flow is being requested. It is in effect when DATAPAUSED in STAT = 1.	
3:2	PAIRCOUNT		Provides the number of I2S channel pairs in this Flexcomm Interface This is a read-only field whose value may be different in other Flexcomm Interfaces. 00 = there is one I2S channel pair in this Flexcomm Interface. 01 = there are two I2S channel pairs in this Flexcomm Interface. 10 = there are three I2S channel pairs in this Flexcomm Interface. 11 = there are four I2S channel pairs in this Flexcomm Interface.	0x3
5:4	MSTSLVCFG		Master / slave configuration selection, determining how SCK and WS are used by all channel pairs in this Flexcomm Interface.	0
		0x0	Normal slave mode, the default mode. SCK and WS are received from a master and used to transmit or receive data.	
		0x1	WS synchronized master. WS is received from another master and used to synchronize the generation of SCK, when divided from the Flexcomm Interface function clock.	
		0x2	Master using an existing SCK. SCK is received and used directly to generate WS, as well as transmitting or receiving data.	
		0x3	Normal master mode. SCK and WS are generated so they can be sent to one or more slave devices.	

Table 697. Configuration register 1 (CFG1, offset = 0xC00) *…continued*

Bit	Symbol	Value	Description	Reset value
7:6	MODE		Selects the basic I2S operating mode. Other configurations modify this to obtain all supported cases. See Section 37.8.2 “Formats and modes” for examples.	0
		0x0	I2S mode a.k.a. “classic” mode. WS has a 50% duty cycle, with (for each enabled channel pair) one piece of left channel data occurring during the first phase, and one pieces of right channel data occurring during the second phase. In this mode, the data region begins one clock after the leading WS edge for the frame. **Remark:** For a 50% WS duty cycle, FRAMELEN must define an even number of I2S clocks for the frame. If FRAMELEN defines an odd number of clocks per frame, the extra clock will occur on the right.	
		0x1	DSP mode where WS has a 50% duty cycle. See Remark for MODE 0.	
		0x2	DSP mode where WS has a one clock long pulse at the beginning of each data frame.	
		0x3	DSP mode where WS has a one data slot long pulse at the beginning of each data frame.	
8	RIGHTLOW		Right channel data is in the Low portion of FIFO data. Essentially, this swaps left and right channel data as it is transferred to or from the FIFO. This bit is not used if the data width is greater than 24 bits or if PDMDATA = 1. Note that if the ONECHANNEL field (bit 10 of this register) = 1, the one channel to be used is the nominally the left channel. POSITION can still place that data in the frame where right channel data is normally located. **Remark:** If all enabled channel pairs have ONECHANNEL = 1, then RIGHTLOW = 1 is not allowed.	0
		0	The right channel is taken from the high part of the FIFO data. For example, when data is 16 bits, FIFO bits 31:16 are used for the right channel.	
		1	The right channel is taken from the low part of the FIFO data. For example, when data is 16 bits, FIFO bits 15:0 are used for the right channel.	
9	LEFTJUST		Left justify data.	0
		0	Data is transferred between the FIFO and the I2S serializer/deserializer right justified, i.e. starting from bit 0 and continuing to the position defined by DATALEN. It would correspond to right justified data in the stream on the data bus.	
		1	Data is transferred between the FIFO and the I2S serializer/deserializer left justified, i.e. starting from the MSB of the FIFO entry and continuing for the number of bits defined by DATALEN. It would correspond to left justified data in the stream on the data bus.	
10	ONECHANNEL		Single channel mode. Applies to both transmit and receive. This configuration bit applies only to the first I2S channel pair. Other channel pairs may select this mode independently in their separate CFG1 registers.	0
		0	I2S data for this channel pair is treated as left and right channels.	
		1	I2S data for this channel pair is treated as a single channel, functionally the left channel for this pair. **Remark:** In mode 0 only, the right side of the frame begins at POSITION = 0x100. It is because mode 0 makes a clear distinction between the left and right sides of the frame. When ONECHANNEL = 1, the single channel of data may be placed on the right by setting POSITION to 0x100 + the data position within the right side, for example: 0x108 would place data starting at the 8th clock after the middle of the frame. In other modes, data for the single channel of data is placed at the clock defined by POSITION.	

Table 697. Configuration register 1 (CFG1, offset = 0xC00) *…continued*

Bit	Symbol	Value	Description	Reset value
11	-		Reserved.	-
12	SCK_POL		SCK polarity.	0
		0	Data is launched on SCK falling edges and sampled on SCK rising edges (standard for I2S).	
		1	Data is launched on SCK rising edges and sampled on SCK falling edges.	
13	WS_POL		WS polarity.	0
		0	Data frames begin at a falling edge of WS (standard for classic I2S).	
		1	WS is inverted, resulting in a data frame beginning at a rising edge of WS (standard for most *non-classic* variations of I2S).	
15:14	-		Reserved. Read value is undefined, only zero should be written.	-
20:16	DATALEN		Data length, minus 1 encoded, defines the number of data bits to be transmitted or received for all I2S channel pairs in this Flexcomm Interface. Note that data is only driven to or received from SDA for the number of bits defined by DATALEN. DATALEN is also used in these ways by the I2S: 1. Determines the size of data transfers between the FIFO and the I2S serializer/deserializer. See Section 37.8.4 “FIFO buffer configurations and usage” 2. In mode 1, 2, and 3, determines the location of right data following left data in the frame. 3. In mode 3 (where WS has a one data slot long pulse at the beginning of each data frame) determines the duration of the WS pulse. Values: 0x00 to 0x02 = not supported 0x03 = data is 4 bits in length 0x04 = data is 5 bits in length ... 0x1F = data is 32 bits in length	0
31:21	-	-	Reserved. Read value is undefined, only zero should be written.	-

37.7.2 Configuration register 2

The CFG2 register contains bits that control various aspects of data configuration.

Table 698. Configuration register 2 (CFG2, offset = 0xC04)

Bit	Symbol	Description	Reset value
8:0	FRAMELEN	Frame length, minus 1 encoded, defines the number of clocks and data bits in the frames that this channel pair participates in. See Section 37.8.2.1 “Frame format”. 0x000 to 0x002 = not supported 0x003 = frame is 4 bits in total length 0x004 = frame is 5 bits in total length ... 0x1FF = frame is 512 bits in total length **Remark:** If FRAMELEN is an defines an odd length frame (e.g. 33 clocks) in MODE 0 or 1, the extra clock appears in the right half. **Remark:** When MODE = 3, FRAMELEN must be larger than DATALEN in order for the WS pulse to be generated correctly.	0
15:9	-	Reserved. Read value is undefined, only zero should be written.	-
24:16	POSITION	Data position. Defines the location within the frame of the data for this channel pair. POSITION + DATALEN must be less than FRAMELEN. See Section 37.8.2.1 “Frame format”. **Remark:** When MODE = 0, POSITION defines the location of data in both the left phase and right phase, starting one clock after the WS edge. In other modes, POSITION defines the location of data within the entire frame. ONECHANNEL = 1 while MODE = 0 is a special case, see the description of ONECHANNEL. **Remark:** The combination of DATALEN and the POSITION fields of all channel pairs must be made such that the channels do not overlap within the frame. 0x000 = data begins at bit position 0 (the first bit position) within the frame or WS phase. 0x001 = data begins at bit position 1 within the frame or WS phase. 0x002 = data begins at bit position 2 within the frame or WS phase. ...	0
31:25	-	Reserved. Read value is undefined, only zero should be written.	-

37.7.3 Status register

The STAT register provides status flags for the I2S function, and does not include FIFO status. Note that the FIFO status register supplies peripheral interrupt notification and would be the status register normally observed first for an interrupt service. Some information in this register is read-only, some flags can be cleared by writing a 1 to them, details can be found in Table 699.

Table 699. Status register (STAT, offset = 0xC08)

Bit	Symbol	Value	Description	Reset value	Type
0	BUSY		Busy status for the primary channel pair. Other BUSY flags may be found in the STAT register for each channel pair.	0	RO
		0	The transmitter/receiver for channel pair is currently idle.		
		1	The transmitter/receiver for channel pair is currently processing data.		

Table 699. Status register (STAT, offset = 0xC08) *…continued*

Bit	Symbol	Value	Description	Reset value	Type
1	SLVFRMERR		Slave frame error flag. This applies when at least one channel pair is operating as a slave. An error indicates that the incoming WS signal did not transition as expected due to a mismatch between FRAMELEN and the actual incoming I2S stream.	0	W1C
		0	No error has been recorded.		
		1	An error has been recorded for some channel pair that is operating in slave mode. Error is cleared by writing a 1 to this bit position.		
2	LR		Left/Right indication. This flag is considered to be a debugging aid and is not expected to be used by an I2S driver. Valid when one channel pair is busy. Indicates left or right data being processed for the currently busy channel pair.	-	RO
		0	Left channel.		
		1	Right channel.		
3	DATAPAUSE		Data paused status flag. Applies to all I2S channels	0	RO
		0	Data is not currently paused. A data pause may have been requested but is not yet in force, waiting for an allowed pause point. Refer to the description of the DATAPAUSE control bit in the CFG1 register.		
		1	A data pause has been requested and is now in force.		
31:4	-		Reserved. Read value is undefined, only zero should be written.	-	-

37.7.4 Clock divider register

The DIV register controls how the Flexcomm Interface function clock is used. See Section 37.8.3 “Data rates” for more details.

Remark: DIV must be set to 0 if SCK is used as an input clock for the I2S function, which is the case when the MSTSLVCFG field in the CFG1 register = 0 or 2.

Table 700. Clock divider register (DIV, offset = 0xC1C)

Bit	Symbol	Description	Reset value
11:0	DIV	This field controls how this I2S block uses the Flexcomm Interface function clock. 0x000 = The Flexcomm Interface function clock is used directly. 0x001 = The Flexcomm Interface function clock is divided by 2. 0x002 = The Flexcomm Interface function clock is divided by 3. ... 0xFFF = The Flexcomm Interface function clock is divided by 4,096.	0
31:12	-	Reserved. Read value is undefined, only zero should be written.	-

37.7.5 FIFO configuration register

This register configures FIFO usage. A peripheral must be selected within the Flexcomm Interface prior to configuring the FIFO.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time. Also note that the FIFO for the selected I2S data direction must be enabled because the FIFO is the only means for accessing I2S data.

Table 701. FIFO configuration register (FIFOCFG, offset = 0xE00)

Bit	Symbol	Value	Description	Reset value	Access
0	ENABLETX		Enable the transmit FIFO.	0	R/W
		0	The transmit FIFO is not enabled.		
		1	The transmit FIFO is enabled.		
1	ENABLERX		Enable the receive FIFO.	0	R/W
		0	The receive FIFO is not enabled.		
		1	The receive FIFO is enabled.		
2	TXI2SE0		Transmit I2S empty 0.Determines the value sent by the I2S in transmit mode if the TX FIFO becomes empty. This value is sent repeatedly until the I2S is paused, the error is cleared, new data is provided, and the I2S is un-paused.	0	R/W
		0	If the TX FIFO becomes empty, the last value is sent. This setting may be used when the data length is 24 bits or less, or when MONO = 1 for this channel pair.		
		1	If the TX FIFO becomes empty, 0 is sent. Use if the data length is greater than 24 bits or if zero fill is preferred.		
3	PACK48		Packing format for 48-bit data. it relates to how data is entered into or taken from the FIFO by software or DMA.	0	R/W
		0	48-bit I2S FIFO entries are handled as all 24-bit values.		
		1	48-bit I2S FIFO entries are handled as alternating 32-bit and 16-bit values.		
5:4	SIZE		FIFO size configuration. It is a read-only field. 0x0, 0x1 = not applicable to I2S. 0x2 = FIFO is configured as eight entries of 32 bits, each corresponding to two 16-bit data values for left and right channels. This setting occurs when the I2S DATALEN is less than 16 bits, or from 25 to 32 bits. 0x3 = FIFO is configured as 8 entries of 48 bits, each corresponding to either 2 16-bit data values for left and right channels. This setting occurs when the I2S DATALEN is from 17 to 24 bits.	-	RO
11:6	-		Reserved. Read value is undefined, only zero should be written.	-	-
12	DMATX		DMA configuration for transmit.	0	R/W
		0	DMA is not used for the transmit function.		
		1	Generate a DMA request DMA for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.		
13	DMARX		DMA configuration for receive.	0	R/W
		0	DMA is not used for the receive function.		
		1	Generate a DMA request DMA for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.		

Table 701. FIFO configuration register (FIFOCFG, offset = 0xE00) *…continued*

Bit	Symbol	Value	Description	Reset value	Access
14	WAKETX		Wake-up for transmit FIFO level. it allows the device to be woken from reduced power modes up to deep-sleep, as long as the peripheral function works in that power mode, without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.	0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.		
15	WAKERX		Wake-up for receive FIFO level. It allows the device to be woken from reduced power modes up to deep-sleep, as long as the peripheral function works in that power mode, without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.	0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.		
16	EMPTYTX		Empty command for the transmit FIFO. When a 1 is written to this bit, the TX FIFO is emptied.	-	WO
17	EMPTYRX		Empty command for the receive FIFO. When a 1 is written to this bit, the RX FIFO is emptied.	-	WO
31:18	-		Reserved. Read value is undefined, only zero should be written.	-	-

37.7.6 FIFO status register

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 702. FIFO status register (FIFOSTAT, offset = 0xE04)

Bit	Symbol	Description	Reset value	Access
0	TXERR	TX FIFO error. Will be set if a transmit FIFO error occurs.It can be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit.	0	R/W1C
1	RXERR	RX FIFO error. Will be set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit.	0	R/W1C
2	-	Reserved. Read value is undefined, only zero should be written.	-	-
3	PERINT	Peripheral interrupt. When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral's STAT register.	0	RO
4	TXEMPTY	Transmit FIFO empty. When 1, the transmit FIFO is empty. The peripheral may still be processing the last piece of data.	1	RO
5	TXNOTFULL	Transmit FIFO not full. When 1, the transmit FIFO is not full, so more data can be written. When 0, the transmit FIFO is full and another write would cause it to overflow.	1	RO
6	RXNOTEMPTY	Receive FIFO not empty. When 1, the receive FIFO is not empty, so data can be read. When 0, the receive FIFO is empty.	0	RO
7	RXFULL	Receive FIFO full. When 1, the receive FIFO is full. Data needs to be read out to prevent the peripheral from causing an overflow.	0	RO
12:8	TXLVL	Transmit FIFO current level. A 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. Other values tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, the TXEMPTY and TXNOTFULL flags will be 0.	0	RO
15:13	-	Reserved. Read value is undefined, only zero should be written.	-	-
20:16	RXLVL	Receive FIFO current level. A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. Other values tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, the RXFULL and RXNOTEMPTY flags will be 1.	0	RO
31:21	-	Reserved. Read value is undefined, only zero should be written.	-	-

37.7.7 FIFO trigger settings register

This register allows selecting when FIFO-level related interrupts occur.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 703. FIFO trigger settings register (FIFOTRIG, offset = 0xE08)

Bit	Symbol	Value	Description	Reset value
0	TXLVLENA		Transmit FIFO level trigger enable. The FIFO level trigger will cause an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests. See DMATX in FIFOCFG.	0
		0	Transmit FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.	
1	RXLVLENA		Receive FIFO level trigger enable. This trigger will become an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests. See DMARX in FIFOCFG.	0
		0	Receive FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.	
10:2	-	-	Reserved. Read value is undefined, only zero should be written.	-
11:8	TXLVL		Transmit FIFO level trigger point. This field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. 0 = generate an interrupt when the TX FIFO becomes empty. 1 = generate an interrupt when the TX FIFO level decreases to one entry. ... 7 = generate an interrupt when the TX FIFO level decreases to 7 entries (is no longer full).	0
15:12	-	-	Reserved. Read value is undefined, only zero should be written.	-
19:16	RXLVL		Receive FIFO level trigger point. The RX FIFO level is checked when a new piece of data is received. This field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. 0 = generate an interrupt when the RX FIFO has one entry (is no longer empty). 1 = generate an interrupt when the RX FIFO has two entries. ... 7 = generate an interrupt when the RX FIFO increases to eight entries (has become full).	0
31:20	-	-	Reserved. Read value is undefined, only zero should be written.	-

37.7.8 FIFO interrupt enable set and read

This register is used to enable various interrupt sources. 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 FIFOINTENCLR register is used to clear bits in this register.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 704. FIFO interrupt enable set and read register (FIFOINTENSET, offset = 0xE10)

Bit	Symbol	Value	Description	Reset value
0	TXERR		Determines whether an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register.	0
		0	No interrupt will be generated for a transmit error.	
		1	An interrupt will be generated when a transmit error occurs.	
1	RXERR		Determines whether an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register.	0
		0	No interrupt will be generated for a receive error.	
		1	An interrupt will be generated when a receive error occurs.	
2	TXLVL		Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0
		0	No interrupt will be generated based on the TX FIFO level.	
		1	If TXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.	
3	RXLVL		Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0
		0	No interrupt will be generated based on the RX FIFO level.	
		1	If RXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.	
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

37.7.9 FIFO interrupt enable clear and read

The FIFOINTENCLR register is used to clear interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Table 705. FIFO interrupt enable clear and read (FIFOINTENCLR, offset = 0xE14)

Bit	Symbol	Description	Reset value
0	TXERR	Writing a one to this bit disables the TXERR interrupt.	0x0
1	RXERR	Writing a one to this bit disables the RXERR interrupt.	0x0
2	TXLVL	Writing a one to this bit disables the interrupt caused by the transmit FIFO reaching the level specified by the TXLVL field in the FIFOTRIG register.	0x0
3	RXLVL	Writing a one to this bit disables the interrupt caused by the receive FIFO reaching the level specified by the RXLVL field in the FIFOTRIG register.	0x0
31:4	-	Reserved. Read value is undefined, only zero should be written.	-

37.7.10 FIFO interrupt status register

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are both pending and currently enabled. it can simplify software handling of interrupts. See Section 37.7.6 “FIFO status register” and Section 37.7.7 “FIFO trigger settings register” for description of interrupts details.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 706. FIFO interrupt status register (FIFOINTSTAT, offset = 0xE18)

Bit	Symbol	Description	Reset value
0	TXERR	TX FIFO error.	0
1	RXERR	RX FIFO error.	0
2	TXLVL	Transmit FIFO level interrupt.	0
3	RXLVL	Receive FIFO level interrupt.	0
4	PERINT	Peripheral interrupt.	0
31:5	-	Reserved. Read value is undefined, only zero should be written.	-

37.7.11 FIFO write data register

The FIFOWR register is used to write values to be transmitted to the FIFO. Details of how FIFOWR and FIFOWR48H are used can be found in Section 37.8.4 “FIFO buffer configurations and usage”.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 707. FIFO write data register (FIFOWR, offset = 0xE20)

Bit	Symbol	Description	Reset value
31:0	TXDATA	Transmit data to the FIFO. The number of bits used depends on configuration details.	-

37.7.12 FIFO write data for upper data bits

The FIFOWR48H register is used under certain conditions to write values to the FIFO. See Section 37.8.4 “FIFO buffer configurations and usage” for FIFOWR and FIFOWR48H details.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 708. FIFO write data for upper data bits (FIFOWR48H, offset = 0xE24)

Bit	Symbol	Description	Reset value
23:0	TXDATA	Transmit data to the FIFO. Whether this register is used and the number of bits used depends on configuration details.	-
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

37.7.13 FIFO read data register

The FIFORD register is used to read values that have been received by the FIFO. See Section 37.8.4 “FIFO buffer configurations and usage” for FIFORD and FIFORD48H details.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 709. FIFO read data register (FIFORD, offset = 0xE30)

Bit	Symbol	Description	Reset value
31:0	RXDATA	Received data from the FIFO. The number of bits used depends on configuration details.	-

37.7.14 FIFO read data for upper data bits

The FIFORD48H register is used under certain conditions to read values from the FIFO. See Section 37.8.4 “FIFO buffer configurations and usage” for FIFORD and FIFORD48H details.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 710. FIFO read data for upper data bits (FIFORD48H,offset = 0xE34)

Bit	Symbol	Description	Reset value
23:0	RXDATA	Received data from the FIFO. Whether this register is used and the number of bits used depends on configuration details.	-
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

37.7.15 FIFO data read with no FIFO pop

This register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). It can allow system software to observe incoming data without interfering with the peripheral driver.

Table 711. FIFO data read with no FIFO pop (FIFORDNOPOP, offset = 0xE40)

Bit	Symbol	Description	Reset value
31:0	RXDATA	Received data from the FIFO.	-

37.7.16 FIFO data read for upper data bits with no FIFO pop

This register acts in exactly the same way as FIFORD48H, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). It can allow system software to observe incoming data without interfering with the peripheral driver.

Table 712. FIFO data read for upper data bits with no FIFO pop (FIFORD48HNOPOP, offset = 0xE44)

Bit	Symbol	Description	Reset value
23:0	RXDATA	Received data from the FIFO.	-
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

37.7.17 FIFO size register

The FIFOSIZE register provides the size FIFO for the selected Flexcomm function on this device.

Table 713. FIFO size register (FIFOSIZE - offset = 0xE48)

Bit	Symbol	Description	Reset value
4:0	FIFOSIZE	Provides the size of the FIFO for software. The size of the I2S FIFO is 8 entries.	0x08
31:5	-	Reserved.	-

37.7.18 Module identification register

The ID register identifies the type and revision of the module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 714. Module identification register (ID, offset = 0xFFC)

Bit	Symbol	Description	Reset value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x00
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE090

37.8 Functional description

37.8.1 AHB bus access

The bus interface to the I2S registers contained in the Flexcomm Interface support only word writes. Byte and half-word writes are not supported in conjunction with the I2S function.

37.8.2 Formats and modes

The format of data frames and WS is determined by several fields in the CFG1 and CFG2 registers, see Section 37.7.1 “Configuration register 1” and Section 37.7.2 “Configuration register 2” respectively. CFG1 and CFG2 together control the formatting of the data and the format of the frame in which the data is contained.

37.8.2.1 Frame format

The overall frame format is defined by fields in the CFG1 and CFG2 registers. The frame includes data related to the primary channel pair and any other channel pairs implemented by this I2S. These fields plus the position of data for each channel pair, as determined by the POSITION field in CFG2, define the main features of the frame.

- MODE: 2-bit field in CFG1 that defines the overall character of the frame.
- FRAMELEN: 9-bit field in CFG2, defines the length of the data frame this I2S participates in. This field is Minus 1 encoded: the value 63 means 64 clocks and bit positions in each frame.
- DATALEN: 5-bit field in CFG1, defines the number of data bits that are used by the transmitter or receiver. This field is minus 1 encoded: the value 15 means 16 data bits. For each channel pair, data is only driven to or received from SDA for the number of bits defined by DATALEN.

  DATALEN is also used in these ways:

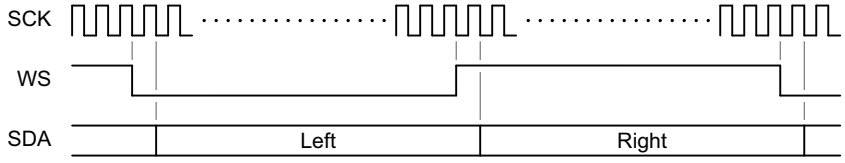
  1) Determines the size of data transfers between the FIFO and the I2S serializer/deserializer.

  2) When MODE = 0x1, 0x2, or 0x3 (i.e. not 0x0), determines the position of right data following left data within the frame.

  3) When MODE = 0x3, determines the duration of the WS pulse.

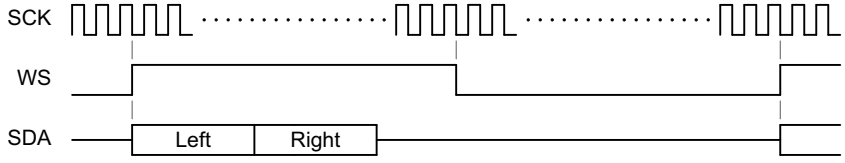
37.8.2.2 Example frame configurations

A sampling of frame slot formats are shown in the following figures. It is not an exhaustive set of possibilities, but shows the various frame formatting concepts. Note that slot identifications are illustrative only, data positions are flexible and there are no predefined slots for the hardware.



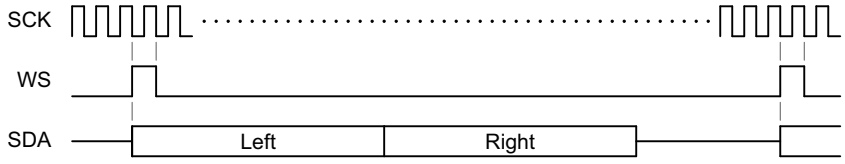
MODE = 0; POSITION = 0; SCK_POL = 0; WS_POL = 0; MONO = 0

Fig 133. Classic I2S mode



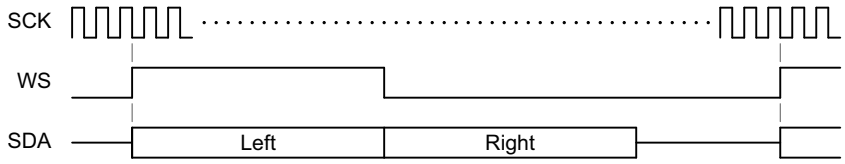
MODE = 1; POSITION = 0; SCK_POL = 0; WS_POL = 1; MONO = 0

Fig 134. DSP mode with 50% WS



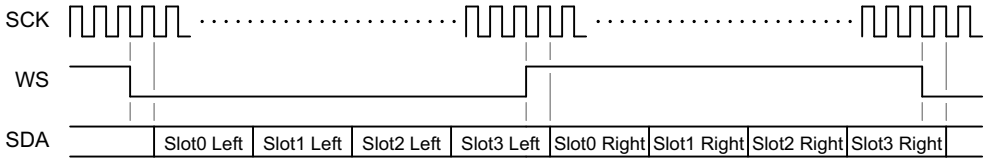
MODE = 2; POSITION = 0; SCK_POL = 0; WS_POL = 1; MONO = 0

Fig 135. DSP mode with 1 SCK pulsed WS



MODE = 3; POSITION = 0; SCK_POL = 0; WS_POL = 1; MONO = 0

Fig 136. DSP mode with 1 slot pulsed WS

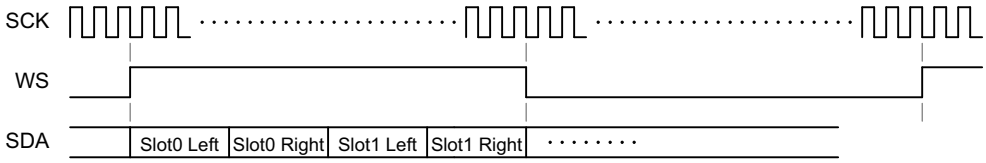


MODE = 0; SCK_POL = 0; WS_POL = 0; MONO = 0

POSITION = bit position of the first used data bit for a slot (within the data for each WS phase).

One Left/Right slot is used by one I2S channel pair. This example shows 4 data slots.

Fig 137. TDM in classic I2S mode

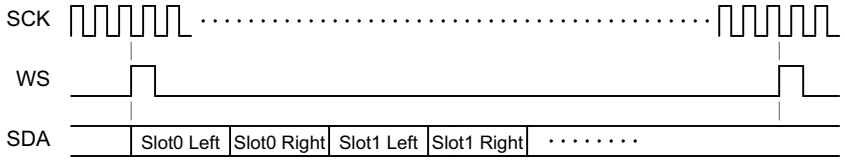


MODE = 1; SCK_POL = 0; WS_POL = 1; MONO = 0

POSITION = bit position of the first used data bit for each slot.

TDM and DSP modes with 50% WS. One Left/Right slot would be used by 1 slice.

Fig 138. TDM and DSP modes with 50% WS

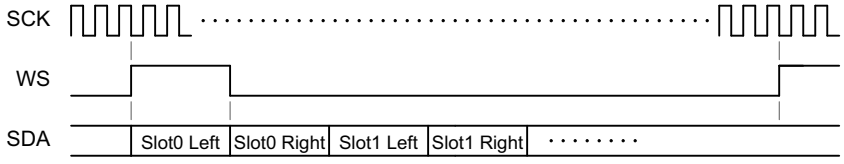


MODE = 2; SCK_POL = 0; WS_POL = 1; MONO = 0

POSITION = bit position of the first used data bit for each slot.

One Left/Right slot would be used by one channel pair.

Fig 139. TDM and DSP modes with 1 SCK pulsed WS

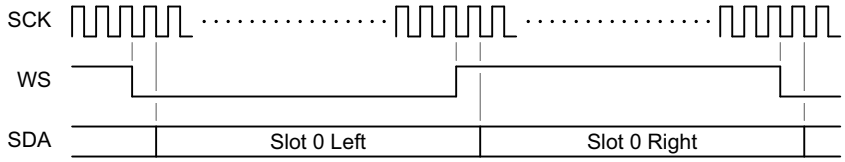


MODE = 3; SCK_POL = 0; WS_POL = 1; MONO = 0

POSITION = bit position of the first used data bit for each slot.

One Left/Right slot would be used by one channel pair.

Fig 140. TDM and DSP modes with 1 slot pulsed WS

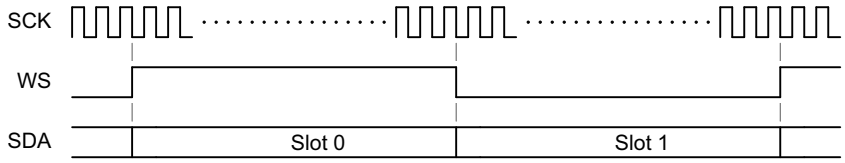


MODE = 0; SCK_POL = 0; WS_POL = 0; MONO = 1.

POSITiON = bit position of the first used data bit for slot 0 Left, bit position within the second half + 0x100 for slot 0 Right.

One slot would be used by one I2S.

Fig 141. I2S mode, mono

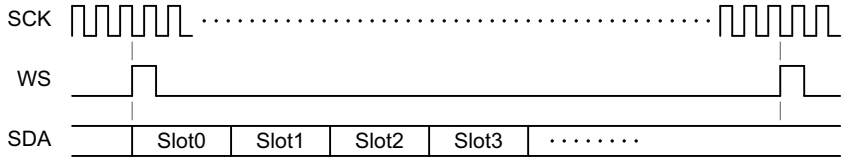


MODE = 1; SCK_POL = 0; WS_POL = 1; MONO = 1

POSITiON = bit position of the first used data bit for each slot.

One slot would be used by one I2S.

Fig 142. DSP mode, mono



MODE = 2; SCK_POL = 0; WS_POL = 1; MONO = 1.

POSITION = bit position of the first used data bit for each slot.

One slot would be used by one I2S.

Fig 143. TDM and DSP modes, mono, with WS pulsed for one SCK time

37.8.2.3 I2S signal polarities

Figure 144 shows examples of SCK and WS polarities and how they relate to data positions.

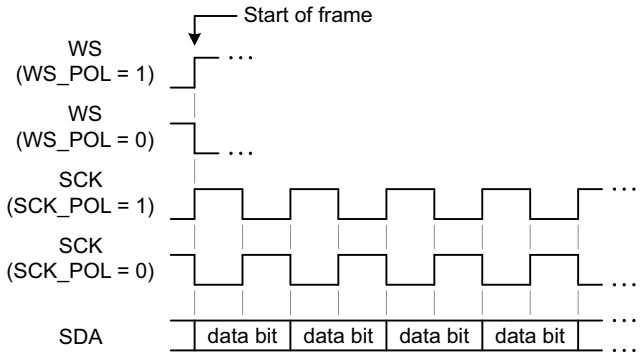


Fig 144. Data at the start of a frame, shown with both SCK and WS polarities

37.8.3 Data rates

37.8.3.1 Rate support

The actual I2S clock rates, sample rates, etc. that can be supported depend on the clocking that is available to run the interface. As a slave, the interface will be receiving SCK from a master. In that case, there is an upper limit to how fast the interface can operate, it will be specified in the interface AC characteristics in a specific device data sheet and a limit to how much data and be transferred across clock domains and handled by the CPU.

In general, the I2S can support:

- Standard sample rates such as 16, 22.05, 32, 44.1, 48, and 96 kHz, and others.
- External MCLK inputs up to approximately 25 MHz (256 fs of a 96 kHz sample rate) and more. Refer to a specific device data sheet for details.

37.8.3.2 Rate calculations

For operation as a master, the frequency need as the clock input of the I2S is generally an integer multiple of:

- Frame/sample rate * number of bits/clocks in a data frame

If this is a multiple of the desired frequency, the I2S function divider can be used to produce the desired frequency.

Example 1

This I2S channel pair is being used to transfer stereo audio data with 32 bit data slots and a 96 kHz sample rate.

Setup: the sample rate is 96 kHz, the frame is configured for two 32-bit data slots (32-bit stereo). The function clock divider output rate would be 96,000 * (2 * 32) = 6.144 MHz.

The value of DIV would be (function clock divider input frequency / the required divider output frequency) - 1. If the divider input is 24.576 MHz (256 fs of the 96 kHz sample rate), the divider needs to divide by 4 (DIV = 3) to obtain the target divider output rate of 6.144 MHz.

Example 2

This I2S channel pair is being used to supply one 16-bit data slot in a 4 slot frame with a frame rate of 50 kHz.

Setup: the sample rate is 50 kHz, the frame is configured four 16-bit data slots, The function clock divider output rate would be 50,000 * (4 * 16) = 3.2 MHz.

The value of DIV would be (function clock divider input frequency / the required divider output frequency) - 1. If the divider input is 16 MHz, the divider needs to divide by 5 (DIV = 4) to obtain the target divider output rate of 3.2 MHz.

37.8.4 FIFO buffer configurations and usage

The Flexcomm Interface supports several possibilities of data packing/unpacking depending on the size of data being handled.

Some details of FIFO usage are determined by the value of the I2S DATALEN field in the CFG1 register, and some other configuration bits as follows:

- If DATALEN specifies a number of data bits from 4 to 16:
  - The FIFO will be configured as 32 bits wide and eight entries deep.
  - Each data transfer between the bus and the FIFO will be a pair of left and right values, which fit into a 32-bit word. The order of left and right data is selectable via the RIGHTLOW configuration bit.
  - If a channel pair is configured with ONECHANNEL = 1, then only one value is transferred, nominally the left.
- If DATALEN specifies a number of data bits from 17 to 24:
  - The FIFO will be configured as 48 bits wide and eight entries deep.
  - Data transfer between the bus and the FIFO depends the PACK48 configuration bit and whether or not DMA is enabled. When DMA is enabled, all transfers are done with FIFOWR or FIFORD. When DMA is not enabled, transfers will alternate between FIFOWR or FIFORD and FIFOWR48H or FIFORD48H, depending on the data direction selected for the I2S function. In all cases, the two transfers will constitute a pair of left and right values. The order of left and right data is selectable via the RIGHTLOW configuration bit.
  - If PACK48 = 0, each of the two transfers both define 17 to 24 bits of data. If PACK48 = 1, the first transfer provides 32 bits of data, the second provides the remainder need to complete the paired data as defined.
  - If a channel pair is configured with ONECHANNEL = 1, then only the left value is transferred using the FIFOWR or FIFORD register.
- If DATALEN specifies a number of data bits from 25 to 32:
  - The FIFO will be configured as 32 bits wide and eight entries deep.
  - Each data transfer between the bus and the FIFO will be a single value, starting with left, then right.

– If a channel pair is configured with ONECHANNEL = 1, then only one value is transferred.

37.8.5 DMA

The Flexcomm Interface can generate DMA requests based on FIFO levels. Data transfers for any channel can be handled by DMA once the I2S clocking and has been configured, that channel has been configured, DMA has been configured, and the I2S bus is running. DMA operation is similar to any other serial peripheral.

DMA related configurations in the Flexcomm Interface I2S may be found in the FIFOCFG register, see Section 37.7.5 “FIFO configuration register” ,bits DMATX, DMARX, WAKETX, WAKERX, and PACK48, and in the FIFOTRIG register, see Section 37.7.7 “FIFO trigger settings register” bits TXLVLENA, RXLVLENA, and fields TXLVL and RXLVL.

37.8.6 Clocking and power considerations

The master function of the I2S requires the Flexcomm Interface function clock to be running in order to operate. The slave function can operate using external clocks, and can wake up the CPU when data is needed or available.

38.1 How to read this chapter

The PLU is available on all parts.

38.2 Features

- The Programmable Logic Unit is used to create small combinatorial and/or sequential logic networks including simple state machines.
- The PLU is comprised of an array of 26 inter-connectable, 5-input Look-up Table (LUT) elements, and four flip-flops.
- Eight primary outputs can be selected using a multiplexer from among all of the LUT outputs and the four flip-flops.
- An external clock to drive the four flip-flops must be applied to the PLU_CLKIN pin if a sequential network is implemented.
- Programmable logic can be used to drive on-chip inputs/triggers through external pin-to-pin connections.
- A tool suite is provided to facilitate programming of the PLU to implement the logic network described in a Verilog RTL design.

38.3 Pin description

There are up to six primary inputs into the PLU module, one clock input, and eight primary outputs. All the inputs are connected directly to the package pins via chip-level I/O multiplexing. All these pins can be enabled by configuring the relevant SWM register.

A particular logic network may not require all of the available inputs or outputs. The user can specify which inputs and outputs to use, and which package pins those inputs and outputs will connect to as part of the overall top-level IO configuration.

If the logic network utilizes one or more of the four “state” flip-flops, an external clock must be applied to the PLU_CLKIN input. The package pin used for this function is specified using the top-level I/O multiplexing of the chip. All other PLU inputs must meet specified setup and hold times relative to this clock input. Output timing is also specified relative to this pin.

If the logic network is purely combinatorial, there is no need to provide an input clock to PLU_CLKIN.

Table 715. Time interval register (INTVAL[0:3], offset = 0x000 (INTVAL0) to 0x030 (INTVAL3))

Bit	Description
PLU_IN [5:0]	Primary inputs. All plu_inputs are available as input sources to all the LUT elements.
PLU_CLKIN	Input clock to the four “state” flip-flops, if used. Not required for purely combinatorial networks. Input/Output timing specified relative to this clock.
PLU_OUT [7:0]	Primary outputs. Selectable via multiplexers from among all LUT element outputs and the four “state” flip-flops.

38.4 General description

The PLU is comprised of 26 5-input LUT elements. Each LUT element contains a 32-bit truth table (look-up table) register and a 32:1 multiplexer. During operation, the five LUT inputs control the select lines of the multiplexer. This structure allows any desired logical combination of the five LUT inputs.

The five inputs to each LUT can be driven from a selection of sources comprised of primary inputs from pins, together with the outputs from all of the other LUTs and the outputs of the four *state* flip-flops. A set of multiplexers associated with each LUT is used to select the five inputs to that LUT. These multiplexers are controlled by registers, which are programmed during initialization. Connecting multiple LUT elements together permits construction of complex boolean expressions.

The outputs of up to four of the LUTs can be captured in one of the four *state* flip-flops, which can then be used as primary outputs and/or connected to the inputs of other LUTs. These four flip-flops, if used by the target logic network, are clocked by the external *plu_clkin* supplied by the user.

Note: The four *state* flip-flops are automatically cleared to ‘0’ by the internal chip reset signal. If a different initial state is required than all-zeros for these flip-flops, the user must force the primary inputs at start-up to some combination that will achieve the required state. That start-up combination must be maintained through at least one plu_clkin rising-edge.

There are eight primary output pins. Any of the ‘n’ LUT outputs or the four flip-flops can be selected to construct the eight primary outputs.

Remark: In general, once the PLU module is configured, the PLU bus clock can be shut-off to conserve power. The only exception to this is when there is a need to read the outputs register while the PLU is operational. In that case, the PLU bus clock must be re-enabled prior to performing the read.

Figure 145 shows the PLU block diagram with the following configuration: 26 LUT elements; six primary inputs.

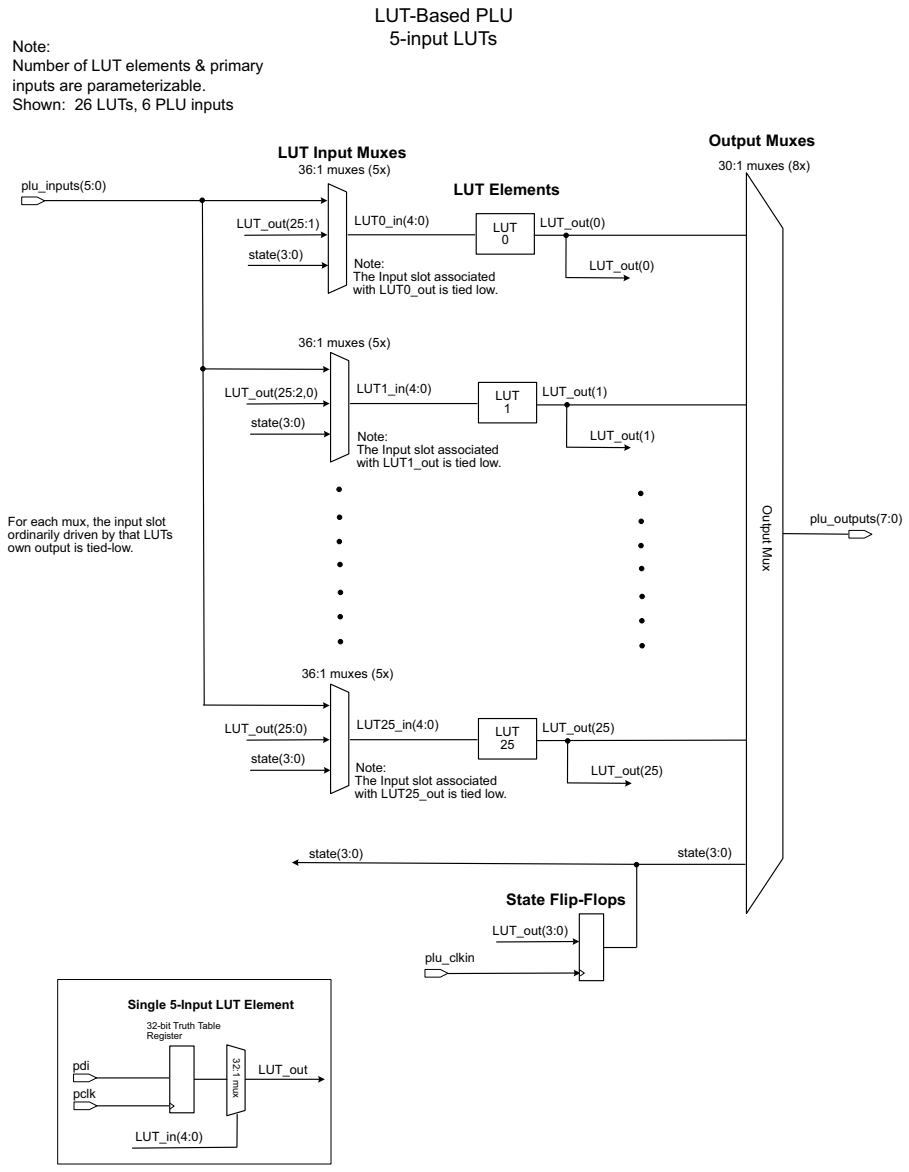


Fig 145. PLU block diagram

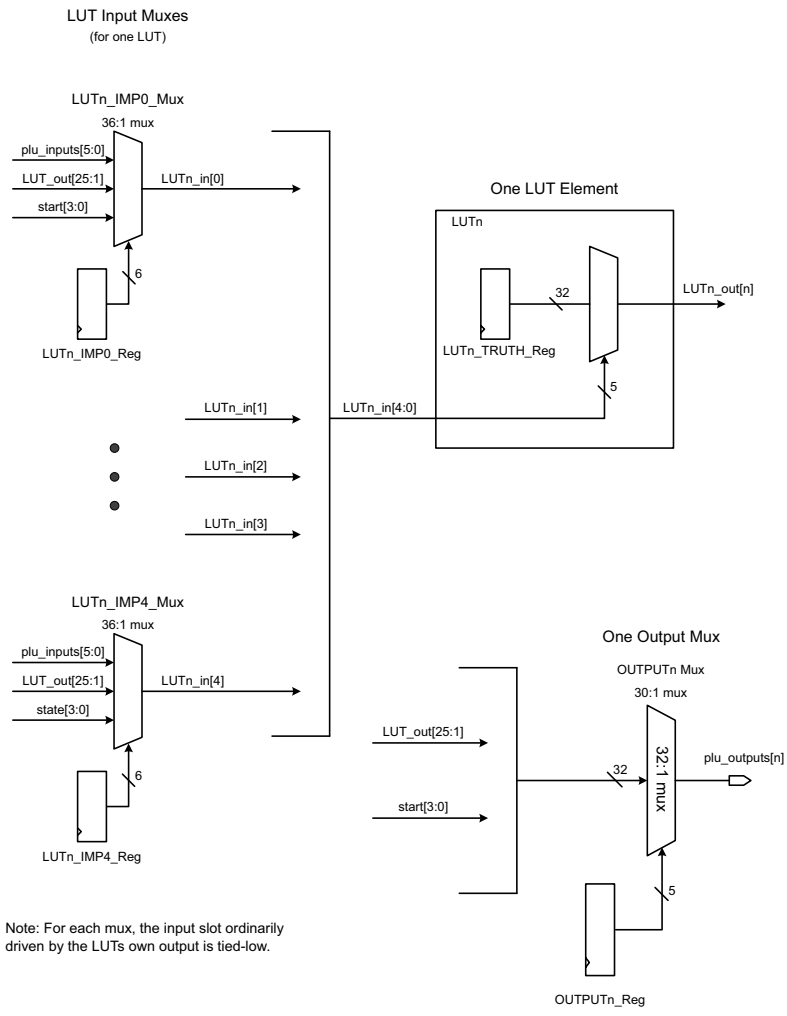


Fig 146. PLU register/multiplexing detail

38.4.1 Using the Programmable Logical Unit

Programming the PLU to implement a particular logic network involves writing to the various truth table registers to specify the logic functions to be performed by each of the LUT elements, programming the Input multiplexer registers to select the five inputs presented to each LUT, and programming the output multiplexer register to select the eight primary outputs from the PLU module.
Programming of all of these registers is performed during initialization. Do not write to the registers during operation.

To facilitate programming of the PLU, a tool suite is provided. The tools report the values that must be written to all the registers in Table 716 to implement a logic network described in a Verilog RTL input file. See Section 38.4.2 “Description of tool flow” for a complete description of the tool flow.

Programming the I/O multiplexing through the SWM is needed to connect the required number of PLU primary inputs and outputs to pins. See Chapter 16 “LPC55S6x/LPC55S2x/LPC552x General Purpose I/O (GPIO)”.

38.4.2 Description of tool flow

The PLU programming tool suite that is provided facilitates configuring the PLU to implement the desired logic network. The input to the tool is a Verilog RTL description of the functionality to be implemented.

The output of the tool suite provides the following

1. Values to be programmed into all LUT INPUT MUX registers.
2. Values to be programmed into all LUT TRUTH registers.
3. Values to be programmed into all OUTPUT MUX registers.
4. Error response if the described network cannot be implemented.
   (most likely due to an excessive amount of logic or use of more than four flip-flops)
5. AC Timing parameters for the implemented design.

38.5 Register description

There are 26 LUT elements in the PLU. For each LUT element there are five Input Multiplexer registers and one truth-table (*Look-up Table*) register.

The PLU has eight output multiplexer registers.

Each LUT input has 35 possible input sources to choose from. Since six bits are required to encode 35 choices, all of the input multiplexer registers are six bits wide. The 26 MSBs for each of these registers should be considered reserved.

All of the registers shown are clocked by the internal bus clock - not the plu_clkin pin. Only the four *state* flip-flops used as part of the target logic network use the externally applied plu_clkin.

Table 716. Register overview: PLU (base address 0x5003D000)

Name	Access	Offset	Description	Reset value	Section
LUTn_INPx_MUX	R/W	0x000-0x010, 0x020-0x030, 0x040-0x050 ... 0x320-0x330	Input select register for LUTn (0 to 25), Inputx (0 to 4) As an example, register offsets for: LUT0_INP0_MUX is 0x000 LUT0_INP1_MUX is 0x004 LUT0_INP2_MUX is 0x008 LUT0_INP3_MUX is 0x00C LUT0_INP4_MUX is 0x010 LUT1_INP0_MUX is 0x020 LUT1_INP1_MUX is 0x024 LUT1_INP2_MUX is 0x028 LUT1_INP3_MUX is 0x02C LUT1_INP4_MUX is 0x030	All 1s	38.5.1
LUTn_TRUTH	R/W	0x800, 0x804, 0x80C ... 0x8FC	Truth-Table (*Look-up Table*) programming for LUTn (0 to 25). As an example, register offsets for LUT0_TRUTH is 0x800 LUT1_TRUTH is 0x804 LUT2_TRUTH is 0x808 LUT3_TRUTH is 0x80C LUT4_TRUTH is 0x810	0x0	38.5.2
OUTPUTS	RO	0x900	PLU outputs register (Read-only).	0x0	38.5.4
WAKEINT_CTRL	R/W	0x904	Wake-up/interrupt control.	0x0	38.5.5
OUTPUT0_MUX	R/W	0xC00	Select register for PLU output0.	0x1F	38.5.3
OUTPUT1_MUX	R/W	0xC04	Select register for PLU output1.	0x1F	38.5.3
OUTPUT2_MUX	R/W	0xC08	Select register for PLU output2.	0x1F	38.5.3
OUTPUT3_MUX	R/W	0xC0C	Select register for PLU output3.	0x1F	38.5.3
OUTPUT4_MUX	R/W	0xC10	Select register for PLU output4.	0x1F	38.5.3
OUTPUT5_MUX	R/W	0xC14	Select register for PLU output5.	0x1F	38.5.3
OUTPUT6_MUX	R/W	0xC18	Select register for PLU output6.	0x1F	38.5.3
OUTPUT7_MUX	R/W	0xC1C	Select register for PLU output7.	0x1F	38.5.3

[1] For the LUTn_INP and OUTPUT mux registers, all ones in the implemented bits (lsb's) means none of the input options is selected. In this case the associated multiplexor output is a fixed logic '0'. Typically these registers must be programmed to some valid value..

38.5.1 PLU LUT input multiplexer registers

Each LUT has five inputs and a selection of input sources that can be connected to each of those inputs. These registers control the multiplexers to select which input sources to connect to each LUT input.

Remark: The values that must be programmed into each of these registers is provided by the tool suite.

Table 717. PLU LUT input Mux registers (LUTn_INPx_MUX) address 0x5003D000, 0x000-0x010, 0x020-0x030, 0x040-0x050, ... 0x320-0x330

Bit	Symbol	Description	Reset value
5:0	LUTn_INPx	Selects the input source to be connected to LUTn Input x For each LUT input the available input sources are, in sequence: 1. The PLU primary inputs, beginning with plu_inputs(0). 2. The outputs from all of the other LUT elements (aside from LUTn itself) in order from the lowest to highest-numbered remaining LUTs. (For each LUT, the slot associated with the output from LUTn itself is tied low). 3. The four *state* flip-flops, beginning with state(0). 0x0 programmed in this field will select plu_inputs(0) as the source of this LUT input. Each higher binary value will select one of the other sources in the above list in the order shown. The reset value of *all ones* causes a fixed '0' to be passed to this LUT input. **Remark:** Note that the total number of bits *m* in this field is variable based on the total number of available input sources which, in turn is dependent on the number of primary PLU inputs and the number of LUTs	All 1s
31:6	Reserved	Software should not write ones to reserved bits.	0x0

38.5.2 PLU LUT truth table registers

Each LUT element contains one 32-bit truth table (*Look-up Table*)) register which specifies whether the LUT will output a '0' or a '1' for each combination of the 5 LUT inputs. In other words, these registers specify the complex Boolean expression each individual LUT is to perform

Remark: The values that must be programmed into each of these registers is provided by the tool suite.

Table 718. PLU LUT truth table registers (LUTn_TRUTH) address0x5003D000, 0x800, 0x804, 0x80C 0x8FC

Bit	Symbol	Description	Reset value
31:0	LUTn_TRUTH	Specifies the truth table contents for LUTn..	0

38.5.3 PLU output multiplexer registers

The eight PLU module outputs are specified using these eight registers.

The available choices to comprise the eight PLU outputs are all of the individual LUT element outputs and the four *state* flip-flop outputs.

Remark: The values that must be programmed into each of these registers is provided by the tool suite.

Table 719. PLU output MUX registers (PLU_OUTPUTn_MUX, address = 0x5003D000, 0xC00-0xC1C)

Bit	Symbol	Description	Reset value
4:0	OUTPUTn	Selects the source to be connected to PLU Output n For each LUT input the available input sources are, in sequence: 1. The outputs from all of the available LUT elements, beginning with LUT0. 2. The four *state* flip-flop, beginning with state(0). 0x0 programmed in this field will select LUT0 as the source of this output. Each higher binary value will select one of the other sources in the above list in the order shown **Remark:** Note that the total number of bits “m” in this field is variable based on the total number of LUTs provided.	0x1F
31:5	Reserved	Software should not write ones to reserved bits.	

Remark: Note that the eight PLU outputs can only be routed to GPIO pins via SWM. There is no provision to internally connect outputs from the programmable logic to on-chip resources such as interrupts, ADC triggers, SCT inputs, and Timer-Capture inputs. Driving these inputs from the programmable logic can be accomplished by externally connecting a PLU output pin to the desired GPIO input pin.

38.5.4 PLU outputs register

The eight selected PLU outputs can be read via this read-only register address.

Remark: There is no guarantee that a read of this register will not capture transitional data because of the asynchronous nature of the PLU. It is strongly recommended to read this data multiple times until a consistent result is returned unless it is known that the PLU inputs will be stable during the read operation.

Table 720. PLU outputs register (PLU_OUTPUTS address = 0x5003D000, 0x900)

Bit	Symbol	Description	Reset value
7:0	OUTPUT	Provides the current state of the eight designated PLU outputs. (All 8 bits are available to be read regardless of whether or not they are routed to package pins).	0x0
31:8	Reserved	Software should not write ones to reserved bits.	0x0

38.5.5 Wake-up/interrupt control register

Any of the eight selected PLU outputs can be enabled to contribute to an asynchronous wake-up or an interrupt request. All enabled output signals are logically OR’d together to generate a single wake-up or /Interrupt request.

Remark: There are long and widely disparate delays through the network of LUTs making up the PLU. Therefore, the raw wake-up or interrupt output generated by this logic is prone to glitching (with glitch pulse-widths potentially in the tens of nanoseconds or longer). If used directly, this raw output is likely to result in the generation of spurious wake ups or interrupt requests.

Two alternative options are provided that can be used to eliminate these glitches:

- Glitch suppression option A - registered WAKE/IRQ request
  - For applications where a plu_clkin is provided, an option to use a registered version of the wake-up/interrupt is available. When this option is enabled, the raw wake-up/interrupt request will be set on the rising-edge of the plu_clkin whenever the raw request signal is high. This registered signal will be glitch-free.
  - This option has the added advantage that the wake-up/interrupt request will be maintained until cleared by software. It should be noted that there will be a delay of up to 1-1/2 plu_clkin cycles before the write to clear the registered wake-up/interrupt signal takes effect. Software can poll the clear_wakeintr bit to determine when this has occurred. It should also be noted that if the condition which initially caused the wake-up/interrupt is still active after the registered request is cleared, it will be set again on the next rising edge of plu_clkin.
- Glitch suppression option B - programmable glitch filter
  - For applications where no plu_clkin is present, an alternative mechanism for suppressing glitches is provided by means of a programmable, digital glitch filter. This approach has some potential disadvantages. One disadvantage is that it requires leaving on a clock source that will increase power consumption in low-power operating modes. Another is that the glitch filter will inject delay before the wake-up/interrupt request is generated. The specific details of the glitch filter will be somewhat chip-dependent but, typically it will include a selection of two alternative filter clock sources. One clock will be a low-frequency, low-power clock (For example, a 1 MHz low-power oscillator) which can be used during deep-sleep mode if required. The other will be a higher frequency clock (For example, a 12 MHz 16 MHz or 30 MHz FRO) which will be higher-power but provide shorter latency and finer resolution over the filter width.
  - Once a filter clock source is specified, the user can choose to filter-out pulses of one, two or three cycles of the designated filter clock. Pulses up to one clock-period longer than the designated number of cycles may be filtered-out as well. The above implies that selecting a single cycle of a 1 MHz LPOSC clock means that pulses up to 2 uS wide may be filtered-out. Care must be taken to ensure that legitimate logic states are not missed. This selection will also result in a 1-2 uS delay in assertion of the wake-up/interrupt request.

Table 721. Wake-up interrupt control for PLU (WAKEINT_CTRL, offset = 0x904)

Bit	Symbol	Value	Description	Reset value
7:0	MASK		Interrupt mask (which of the 8 PLU outputs contribute to interrupt) A ‘1’ in each bit in this register enables the corresponding PLU Output to contribute to wake-up/interrupt generation. All enabled PLU outputs are OR’d to generate the wake-up/interrupt request. A ‘0’ in any bit of this register blocks the corresponding PLU Output from causing a wake-up/interrupt.	0x0

Table 721. Wake-up interrupt control for PLU (WAKEINT_CTRL, offset = 0x904) *…continued*

Bit	Symbol	Value	Description	Reset value
9:8	FILTER_MODE		Control input of the PLU, add filtering for glitch.	0x0
		0	Bypass mode. Filtering is disabled. The raw wake-up/interrupt will be passed-through.	
		1	Filter 1 clock period. Any pulse duration shorter than one cycle of the designated filter clock will be filtered-out. Pulse widths up to two cycles long may be filtered.	
		2	Filter 2 clock period. Any pulse duration shorter than two cycles of the designated filter clock will be filtered-out. Pulse widths up to three cycles long may be filtered.	
		3	Filter 3 clock period. Any pulse duration shorter than three cycles of the designated filter clock will be filtered-out. Pulse widths up to four cycles long may be filtered.	
11:10	FILTER_CLKSEL		Selects filter clock.	0x0
		0	Selects the 1 MHz low-power oscillator as the filter clock.	
		1	Selects the 12 Mhz FRO as the filter clock.	
		2	Selects a third filter clock source, if provided.	
		3	N/A	
12	LATCH_ENABLE		Latch the interrupt and then it can be cleared with next bit INTR_CLEAR. Setting this bit specifies use of the registered version of the wake-up/interrupt request instead of the raw or glitch-filtered version. This option can only be used if a plu_clkin clock is provided from off-chip. Note: This mode is not compatible with use of the glitch filter. If this bit is set, the FILTER MODE field should be set to “00” (Bypass Mode). If this bit is set, the wake-up/interrupt request will be set on the rising-edge of plu_clkin whenever the raw wake-up/interrupt signal is high. The request must be cleared by software.	0
13	INTR_CLEAR		When using the registered wake-up/interrupt option (LATCH_ENA = ‘1’) writing a ‘1’ to this bit will clear the wake-up/interrupt request flag. Writing a ‘0’ to this bit has no effect. There will be a delay of up to 1.5 plu_clkin clock cycles before this write-to-clear takes effect. At that point this bit will also be cleared. Software can poll this bit to determine when the wake-up/interrupt request has been removed. Note: It is not necessary for the PLU bus clock to be enabled in order to write-to or read-back this bit. (This is not the case for the other bits of this register).	0
31:14	Reserved		Reserved. Software should not write ones to reserved bits.	0x0

39.1 How to read this chapter

The ADC controller is available on all LPC55S6x/LPC55S2x/LPC552x devices.

The 16-bits analog-to-digital converter (ADC) is a successive-approximation ADC designed for operation within an integrated micro-controller system-on-chip.

Note: Hexadecimal values are designated by a preceding 0x, binary values by a preceding 0b, and decimal values have no preceding character.

39.2 Features

- Linear successive approximation algorithm:
  - Differential operation with 16-bit or 13-bit resolution.
  - Single-ended operation with 16-bit or 12-bit resolution.
  - The ADC support simultaneous conversions on two ADC input channels belonging to a differential pair.
- Channel support for up to 10 analog input channels for conversion of external pins and from internal sources:
  - Select external pin inputs paired for conversion as differential channel input.
  - Measurement of on-chip analog sources, temperature sensor or bandgap.
- Configurable analog input sample time.
- Configurable speed options to accommodate operation in low power modes of SoC.
- Trigger detect with up to 16 trigger sources with priority level configuration. Software or hardware trigger option for each.
- 15 command buffers allow independent options selection and channel sequence scanning.
- Automatic compare for less-than, greater-than, within range, or out-of-range with *store on true* and *repeat until true* options.
- Two independent result FIFOs each contains 16 entries. Each FIFO has configurable watermark and overflow detection.
- Interrupt, DMA or polled operation.
- Linearity and gain offset calibration logic

39.3 Basic configuration

Table 722 describes the chip modes that the ADC block supports.

See Section 39.7.4 “Clock operation” for more information.

Remark: For proper ADC operation, the PDEN_AUXBIAS bit in the PDRUNCFG0 register must be set to “0”(powered) and VREF1VENABLE bit in the AUX_BIAS register must be set to "1" (enabled) prior to using ADC peripheral.

Remark: For best ADC performance, force the offset calibration to -16. Set this prior to performing the gain calibration.

Table 722. Chip modes supported by the ADC block

Chip mode	ADC operation
Run	Normal operation.
Stop/Wait	Can continue operating provided the Doze Enable bit (CTRL[DOZEN]) is clear and the ADC is using an external or internal clock source that continues to operate during stop or wait modes. When the DOZEN bit is set the ADC waits for the current averaging iteration/FIFO storage to complete before acknowledging stop or wait mode entry.
Low leakage stop	The Doze Enable (CTRL[DOZEN]) bit is ignored and the ADC waits for the current transfer to complete any pending operation before acknowledging low leakage mode entry.

39.4 Pin description

39.4.1 ADC signal descriptions

The ADC module supports up to 64 analog channel inputs with differential and single-ended conversion options for all channels. See Section 39.4.2 “Analog channel inputs CHnA and CHnB” for mapped channels.The ADC also requires supply and ground connections.

Table 723. ADC signal descriptions

Signal	Description	I/O
VDDA	Analog power supply.	I
VSSA	Analog ground.	I
VREFN	ADC negative reference voltage	I
VREFP	ADC positve reference voltage	I
ADC0_0 - ADC0_N[1]	A-side analog channel inputs.	I
ADC0_0 - ADC0_N[1]	B-side analog channel inputs.	I

[1] N is dependent on SoC configuration. Refer Table 725 “ADC Inputs Selection & ADC programming” for details.

The voltage reference high (VREFH) used by the ADC is supplied from an off-chip source supplied through the VREFP or VDDA pins. VREFL is always from an external pin and must be at the same voltage potential as VSSA.

This instance of the ADC block supports a programmable selection of the Voltage Reference High used for ADC conversions (via the CFG[REFSEL] field). See table Table 724.

Table 724. VREFH selection

VREF	Mapped to
Vrefh2	VDDA pin.
Vrefh3	VREFP pin

39.4.2 Analog channel inputs CHnA and CHnB

The CMDLa[ADCH] and CMDLa[CTYPE] bitfields control selection of paired or individual input channels.

- Each ADC command independently makes a channel and conversion type selection.
- Each ADCH channel selection has an associated A side and an associated B side input.
- Each ADCH pair can optionally be converted in a differential mode but, only limited pairs are intended to be converted as differential channels (adjacent pins that are designed with matched impedance).

Note: In differential mode the A-side analog channel inputs are the positive input and the B-side analog channel inputs are the negative input.

Table 725. ADC Inputs Selection & ADC programming

ADC Channel # CMDL[ADCH]	GPIOs to be configured as Analog Inputs	Inputs Type	ADC Conversion Mode CMDL[CTYPE] - decimal	Description	Inputs Availability
0	"PIO0_23/ADC0_0 (referred as CH0A) & PIO0_16/ADC0_8 (referred as CH0B)"	"External FAST"	0	Single Ended Conversion of CH0A	HLQFP100, VFBGA98, HTQFP64, VFBGA59
			1	Single Ended Conversion of CH0B	
			2	Differential Conversion CH0A(P)-CH0B(N)	
			3	Dual Single Ended Conversion CH0A & CH0B	
1	"PIO0_10/ADC0_1 (referred as CH1A) & PIO0_11/ADC0_9 (referred as CH1B)"	"External FAST"	0	Single Ended Conversion of CH1A	HLQFP100, VFBGA98, HTQFP64, VFBGA59
			1	Single Ended Conversion of CH1B	
			2	Differential Conversion CH1A(P)-CH1B(N)	
			3	Dual Single Ended Conversion CH1A & CH1B	
2	"PIO0_15/ADC0_2 (referred as CH2A) & PIO0_12/ADC0_10 (referred as CH2B)"	"External FAST"	0	Single Ended Conversion of CH2A	HLQFP100, VFBGA98, HTQFP64, VFBGA59
			1	Single Ended Conversion of CH2B	
			2	Differential Conversion CH2A(P)-CH2B(N)	
			3	Dual Single Ended Conversion CH2A & CH2B	
3	"PIO0_31/ADC0_3 (referred as CH3A) & PIO1_0/ADC0_11 (referred as CH3B)"	"External FAST"	0	Single Ended Conversion of CH3A	HLQFP100, VFBGA98, HTQFP64
			1	Single Ended Conversion of CH3B	
			2	Differential Conversion CH3A(P)-CH3B(N)	
			3	Dual Single Ended Conversion CH3A & CH3B	

Table 725. ADC Inputs Selection & ADC programming

ADC Channel # CMDL[ADCH]	GPIOs to be configured as Analog Inputs	Inputs Type	ADC Conversion Mode CMDL[CTYPE] - decimal	Description	Inputs Availability
4	"PIO1_8/ADC0_4 (referred as CH4A) & PIO1_9/ADC0_12 (referred as CH4B)"	"External STANDARD"	0	Single Ended Conversion of CH4A	HLQFP100, VFBGA98
			1	Single Ended Conversion of CH4B	
			2	Differential Conversion CH4A(P)-CH4B(N)	
			3	Dual Single Ended Conversion CH4A & CH4B	
12	NA	"External STANDARD"	0 - Connect to Side A	VDDA - ADC analog supply domain	NA
13	NA	"External STANDARD"	0 - Connect to Side A	Internal ADC bandgap reference (1V)	NA
26	NA	"External STANDARD"	2 - Connect to both Side A and B	Use Differential Conversion, see Section 39.7.6 "Temperature sensor".	NA

39.4.3 Differential Pairs

The following pairs are selectable for differential measurements.

Table 726. Differential Pairs

Side Polarity	ADCH(0)	ADCH(1)	ADCH(2)	ADCH(3)	ADCH(4)
A - Positive	P0_23	P0_10	P0_15	P0_31	P1_8
B - Negative	P0_16	P0_11	P0_12	P1_0	P1_9

39.4.4 Specific channels

Some ADC channels are used to sample specific signals.

Table 727. ADC Specific channels

Channel	Connection	Description
12	VDDA	VDDA
13	BIAS_VREF_1V	BIAS_VREF_1V from aux_bias module
26	TEMP_SENSOR	Temperature sensor

39.5 General description

Figure 147 shows the ADC block diagram.

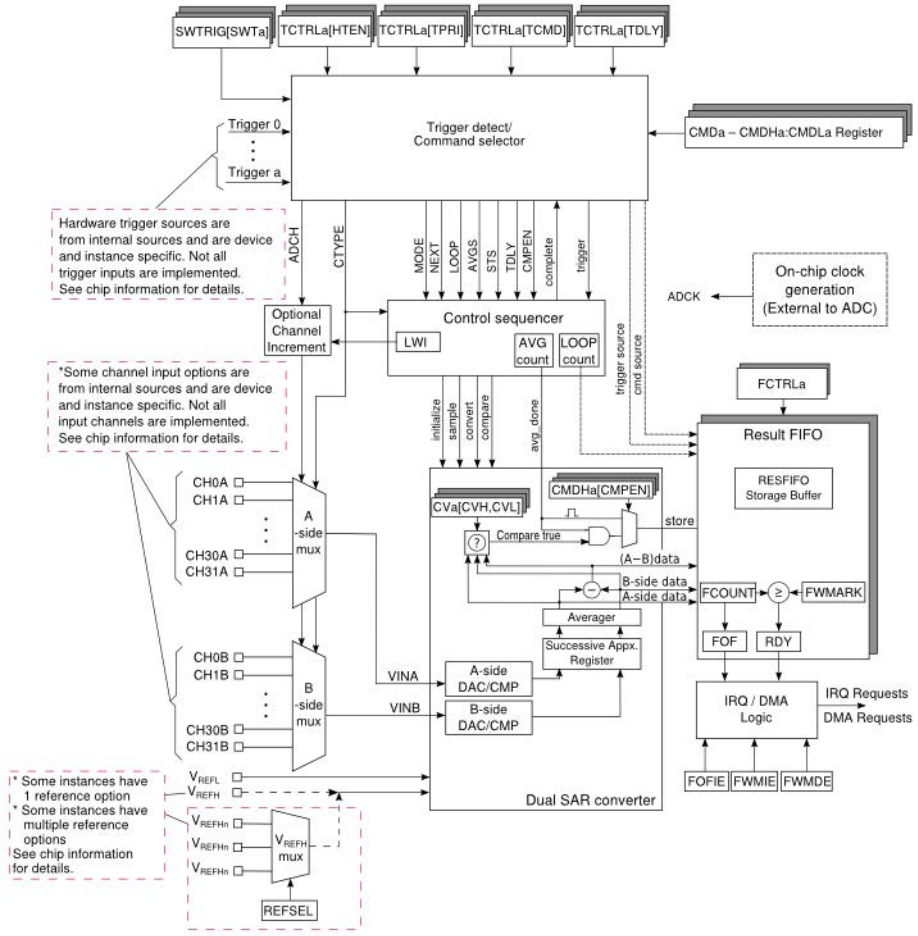


Fig 147. ADC block diagram

The ADC controller provides a great deal of flexibility in launching and controlling sequences of ADC conversions using the associated SAR ADC converter. ADC conversion sequences can be initiated under software control or in response to a selected hardware trigger.

39.6 Register description

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

Table 728. Register overview: base address 0x500A0000

Name	Access	Offset	Description	Reset value	Section
VERID	R	0x0	Version ID register.	0x02002C0B	39.6.1
PARAM	R	0x4	Parameter register.	0x0F041010	39.6.2
CTRL	RW	0x10	ADC control register.	0x0	39.6.3
STAT	RW	0x14	ADC status register.	0x0	39.6.4
IE	RW	0x18	Interrupt enable register.	0x0	39.6.5
DE	RW	0x1C	DMA enable register.	0x0	39.6.6
CFG	RW	0x20	ADC configuration register.	0x00800000	39.6.7
PAUSE	RW	0x24	ADC pause register.	0x0	39.6.8
SWTRIG	WO	0x34	Software trigger register.	0x0	39.6.9
TSTAT	RW	0x38	Trigger status register.	0x0	39.6.10
OFSTRIM	RW	0x40	ADC offset trim register.	0x0	39.6.11
TCTRL0	RW	0xA0	Trigger control register.	0x0	39.6.12
TCTRL1	RW	0xA4	Trigger control register.	0x0	39.6.12
TCTRL2	RW	0xA8	Trigger control register.	0x0	39.6.12
TCTRL3	RW	0xAC	Trigger control register.	0x0	39.6.12
TCTRL4	RW	0xB0	Trigger control register.	0x0	39.6.12
TCTRL5	RW	0xB4	Trigger control register.	0x0	39.6.12
TCTRL6	RW	0xB8	Trigger control register.	0x0	39.6.12
TCTRL7	RW	0xBC	Trigger control register.	0x0	39.6.12
TCTRL8	RW	0xC0	Trigger control register.	0x0	39.6.12
TCTRL9	RW	0xC4	Trigger control register.	0x0	39.6.12
TCTRL10	RW	0xC8	Trigger control register.	0x0	39.6.12
TCTRL11	RW	0xCC	Trigger control register.	0x0	39.6.12
TCTRL12	RW	0xD0	Trigger control register.	0x0	39.6.12
TCTRL13	RW	0xD4	Trigger control register.	0x0	39.6.12
TCTRL14	RW	0xD8	Trigger control register.	0x0	39.6.12
TCTRL15	RW	0xDC	Trigger control register.	0x0	39.6.12
FCTRL0	RW	0xE0	FIFO control register.	0x0	39.6.13
FCTRL1	RW	0xE4	FIFO control register.	0x0	39.6.13
GCC0	R	0xF0	Gain calibration control.	0x0	39.6.14
GCC1	R	0xF4	Gain calibration control.	0x0	39.6.14
GCR0	RW	0xF8	Gain calculation result.	0x0	39.6.15
GCR1	RW	0xFC	Gain calculation result.	0x0	39.6.15
CMDL1	RW	0x100	ADC command low buffer register	0x0	39.6.16
CMDL2	RW	0x108	ADC command low buffer register	0x0	39.6.16
CMDL3	RW	0x110	ADC command low buffer register.	0x0	39.6.16
CMDL4	RW	0x118	ADC command low buffer register	0x0	39.6.16

Table 728. Register overview: base address 0x500A0000 *…continued*

Name	Access	Offset	Description	Reset value	Section
CMDL5	RW	0x120	ADC command low buffer register	0x0	39.6.16
CMDL6	RW	0x128	ADC command low buffer register	0x0	39.6.16
CMDL7	RW	0x130	ADC command low buffer register	0x0	39.6.16
CMDL8	RW	0x138	ADC command low buffer register.	0x0	39.6.16
CMDL9	RW	0x140	ADC command low buffer register	0x0	39.6.16
CMDL10	RW	0x148	ADC command low buffer register	0x0	39.6.16
CMDL11	RW	0x150	ADC command low buffer register.	0x0	39.6.16
CMDL12	RW	0x158	ADC command low buffer register	0x0	39.6.16
CMDL13	RW	0x160	ADC command low buffer register	0x0	39.6.16
CMDL14	RW	0x168	ADC command low buffer register	0x0	39.6.16
CMDL15	RW	0x170	ADC command low buffer register.	0x0	39.6.16
CMDH1	RW	0x104	ADC command high buffer register.	0x0	39.6.17
CMDH2	RW	0x10C	ADC command high buffer register.	0x0	39.6.17
CMDH3	RW	0x114	ADC command high buffer register.	0x0	39.6.17
CMDH4	RW	0x11C	ADC command high buffer register.	0x0	39.6.17
CMDH5	RW	0x124	ADC command high buffer register.	0x0	39.6.17
CMDH6	RW	0x12C	ADC command high buffer register.	0x0	39.6.17
CMDH7	RW	0x134	ADC command high buffer register.	0x0	39.6.17
CMDH8	RW	0x13C	ADC command high buffer register.	0x0	39.6.17
CMDH9	RW	0x144	ADC command high buffer register.	0x0	39.6.17
CMDH10	RW	0x14C	ADC command high buffer register.	0x0	39.6.17
CMDH11	RW	0x154	ADC command high buffer register.	0x0	39.6.17
CMDH12	RW	0x15C	ADC command high buffer register.	0x0	39.6.17
CMDH13	RW	0x164	ADC command high buffer register.	0x0	39.6.17
CMDH14	RW	0x16C	ADC command high buffer register.	0x0	39.6.17
CMDH15	RW	0x174	ADC command high buffer register.	0x0	39.6.17
CV1	RW	0x200	Compare value register.	0x0	39.6.18
CV2	RW	0x204	Compare value register.	0x0	39.6.18
CV3	RW	0x208	Compare value register.	0x0	39.6.18
CV4	RW	0x20C	Compare value register.	0x0	39.6.18
RESFIFO0	R	0x300	ADC data result FIFO register.	0x0	39.6.19
RESFIFO1	R	0x304	ADC data result FIFO register.	0x0	39.6.20
CAL_GAR0	RW	0x400	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR1	RW	0x404	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR2	RW	0x408	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR3	RW	0x40C	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR4	RW	0x410	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR5	RW	0x414	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR6	RW	0x418	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR7	RW	0x41C	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR8	RW	0x420	Calibration general A-side registers.	0x0	39.6.21

Table 728. Register overview: base address 0x500A0000 *…continued*

Name	Access	Offset	Description	Reset value	Section
CAL_GAR9	RW	0x424	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR10	RW	0x428	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR11	RW	0x42C	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR12	RW	0x430	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR13	RW	0x434	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR14	RW	0x438	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR15	RW	0x43C	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR16	RW	0x440	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR17	RW	0x444	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR18	RW	0x448	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR19	RW	0x44C	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR20	RW	0x450	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR21	RW	0x454	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR22	RW	0x458	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR23	RW	0x45C	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR24	RW	0x460	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR25	RW	0x464	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR26	RW	0x468	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR27	RW	0x46C	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR28	RW	0x470	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR29	RW	0x474	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR30	RW	0x478	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR31	RW	0x47C	Calibration general A-side registers.	0x0	39.6.21
CAL_GAR32	RW	0x480	Calibration general A-side registers.	0x0	39.6.21
CAL_GBR0	RW	0x500	Calibration General B-Side registers.	0x0	39.6.22
CAL_GBR1	RW	0x504	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR2	RW	0x508	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR3	RW	0x50C	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR4	RW	0x510	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR5	RW	0x514	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR6	RW	0x518	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR7	RW	0x51C	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR8	RW	0x520	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR9	RW	0x524	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR10	RW	0x528	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR11	RW	0x52C	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR12	RW	0x530	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR13	RW	0x534	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR14	RW	0x538	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR15	RW	0x53C	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR16	RW	0x540	Calibration general B-side registers.	0x0	39.6.22

Table 728. Register overview: base address 0x500A0000 *...continued*

Name	Access	Offset	Description	Reset value	Section
CAL_GBR17	RW	0x544	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR18	RW	0x548	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR19	RW	0x54C	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR20	RW	0x550	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR21	RW	0x554	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR22	RW	0x558	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR23	RW	0x55C	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR24	RW	0x560	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR25	RW	0x564	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR26	RW	0x568	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR27	RW	0x56C	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR28	RW	0x570	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR29	RW	0x574	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR30	RW	0x578	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR31	RW	0x57C	Calibration general B-side registers.	0x0	39.6.22
CAL_GBR32	RW	0x580	Calibration general B-side registers.	0x0	39.6.22

39.6.1 Version ID register

The Version ID register indicates the version integrated for this instance on the device and also indicates inclusion/exclusion of several optional features.

Table 729. Version ID register (VERID, offset = 0x0)

Bit	Symbol	Value	Description	Reset value
0	RES		Resolution.	0x1
		0	Up to 13-bit differential/12-bit single ended resolution supported.	
		1	Up to 16-bit differential/16-bit single ended resolution supported.	
1	DIFFEN		Differential supported.	0x1
		0	Differential operation not supported.	
		1	Differential operation supported. CMDLa[CTYPE] controls fields implemented.	
2	-		Reserved.	0x0
3	MVI		Multi Vref implemented.	0x1
		0	Single voltage reference high (VREFH) input supported.	
		1	Multiple voltage reference high (VREFH) inputs supported.	
6:4	CSW		Channel scale width.	0x0
		0	Channel scaling not supported.	
		1	Channel scaling supported. 1-bit CSCALE control field.	
		6	Channel scaling supported. 6-bit CSCALE control field.	
7	-		Reserved.	0x0
8	VR1RNGI		Voltage reference 1 range control bit implemented.	0x0
		0	Range control not required. CFG[VREF1RNG] is not implemented.	
		1	Range control required. CFG[VREF1RNG] is implemented.	

Table 729. Version ID register (VERID, offset = 0x0) *…continued*

Bit	Symbol	Value	Description	Reset value
9	IADCKI		Internal ADC clock implemented.	0x0
		0	Internal clock source not implemented.	
		1	Internal clock source (and CFG[ADCKEN]) implemented.	
10	CALOFSI		Calibration function Implemented	0x1
		0	Calibration not implemented.	
		1	Calibration implemented.	
11	NUM_SEC		Number of single ended outputs supported.	0x1
		0	This design supports one single ended conversion at a time.	
		1	This design supports two simultaneous single ended conversions.	
14:12	NUM_FIFO		Number of FIFOs.	0x2
		0	N/A	
		1	This design supports one result FIFO.	
		2	This design supports two result FIFOs.	
		3	This design supports three result FIFOs.	
		4	This design supports four result FIFOs.	
15	-		Reserved.	0x0
23:16	MINOR		Minor version number.	0x0
31:24	MAJOR		Major version number.	0x2

39.6.2 Parameter register

The Parameter register indicates the size of several variable integration options for this instance on the device.

Table 730. Parameter Select register (PARAM, offset 0x04)

Bit	Symbol	Value	Description	Reset value
7:0	TRIG_NUM		Trigger number.	0x10
15:8	FIFOSIZE		Result FIFO depth.	0x10
		1	Result FIFO depth = 1 dataword.	
		4	Result FIFO depth = 4 datawords.	
		8	Result FIFO depth = 8 datawords.	
		16	Result FIFO depth = 16 datawords.	
		32	Result FIFO depth = 32 datawords.	
		64	Result FIFO depth = 64 datawords.	
23:16	CV_NUM		Compare value number.	0x4
31:24	CMD_NUM		Command buffer number.	0xF

39.6.3 ADC control register

The ADC control register allows to control the ADC module.

Table 731. ADC control register (CTRL, offset 0x10)

Bit	Symbol	Value	Description	Reset value
0	ADCEN		ADC enable.	0x0
		0	ADC is disabled.	
		1	ADC is enabled.	
1	RST		Software Reset.	0x0
		0	ADC logic is not reset.	
		1	ADC logic is reset.	
2	DOZEN		Doze enable.	0x0
		0	ADC is enabled in Doze mode.	
		1	ADC is disabled in Doze mode.	
3	CAL_REQ		Auto-Calibration request.	0x0
		0	No request for auto-calibration is made.	
		1	A request for auto-calibration is made	
4	CALOFS		Configure for offset calibration function.	0x0
		0	Calibration function disabled.	
		1	Request for offset calibration function.	
7:5	-		Reserved.	0x0
8	RSTFIFO0		Reset FIFO 0	0x0
		0	No effect.	
		1	FIFO 0 is reset.	
9	RSTFIFO1		Reset FIFO 1	0x0
		0	No effect.	
		1	FIFO 1 is reset.	
15:10	-		Reserved.	0x0
18:16	CAL_AVGS		Auto-Calibration Averages.	0x0
		0	Single conversion.	
		1	2 conversions averaged.	
		2	4 conversions averaged.	
		3	8 conversions averaged.	
		4	16 conversions averaged.	
		5	32 conversions averaged.	
		6	64 conversions averaged.	
		7	128 conversions averaged.	
31:19			Reserved.	0x0

39.6.4 ADC status register

The status register provides the current status of the ADC module.

Table 732. ADC status register (STAT, offset = 0x14)

Bit	Symbol	Value	Description	Reset value
0	RDY0		Result FIFO 0 Ready Flag.	0x0
		0	Result FIFO 0 data level not above watermark level.	
		1	Result FIFO 0 holding data above watermark level.	
1	FOF0		Result FIFO 0 Overflow Flag. Write a 0b1 to clear this bit.	0x0
		0	No result FIFO 0 overflow has occurred since the last time the flag was cleared.	
		1	At least one result FIFO 0 overflow has occurred since the last time the flag was cleared.	
2	RDY1		Result FIFO1 ready flag. Write a 0b1 to clear this bit.	0x0
		0	Result FIFO1 data level not above watermark level.	
		1	Result FIFO1 holding data above watermark level.	
3	FOF1		Result FIFO1 overflow flag.	0x0
		0	No result FIFO1 overflow has occurred since the last time the flag was cleared.	
		1	At least one result FIFO1 overflow has occurred since the last time the flag was cleared.	
7:4	-		Reserved.	0x0
8	TEXC_INT		Interrupt flag for high priority trigger exception. Write a 0b1 to clear this bit.	0x0
		0	No trigger exceptions have occurred.	
		1	A trigger exception has occurred and is pending acknowledgment.	
9	TCOMP_INT		Interrupt flag For trigger completion. Write a 0b1 to clear this bit.	0x0
		0	Either IE[TCOMP_IE] is set to 0, or no trigger sequences have run to completion.	
		1	Trigger sequence is completed and all data is stored in the associated FIFO.	
10	CAL_RDY		Calibration ready.	0x0
		0	Calibration is incomplete or not run.	
		1	The ADC is calibrated.	
11	ADC_ACTIVE		ADC active.	0x0
		0	The ADC is IDLE. There are no pending triggers to service and no active commands are being processed.	
		1	The ADC is processing a conversion, running through the power up delay, or servicing a trigger.	
15:12	-		Reserved.	0x0
19:16	TRGACT		Trigger active.	0x0
		0	Command (sequence) associated with trigger 0 currently being executed.	
		1	Command (sequence) associated with trigger 1 currently being executed.	
		2	Command (sequence) associated with trigger 2 currently being executed.	
		0b0011-0b1111	Command (sequence) from the associated trigger number is currently being executed.	

Table 732. ADC status register (STAT, offset = 0x14) *…continued*

Bit	Symbol	Value	Description	Reset value
23:20	-		Reserved.	0x0
27:24	CMDACT		Command active.	0x0
		0	No command is currently in progress.	
		1	Command 1 currently being executed.	
		2	Command 2 currently being executed.	
		0b0011-0b1111	Associated command number is currently being executed.	
31:28	-		Reserved.	0x0

39.6.5 Interrupt enable register

Table 733. Interrupt enable register (IE, offset= 0x18)

Bit	Symbol	Value	Description	Reset value
0	FWMIE0		FIFO 0 watermark interrupt enable.	0x0
		0	FIFO 0 watermark interrupts are not enabled.	
		1	FIFO 0 watermark interrupts are enabled.	
1	FOFIE0		Result FIFO 0 overflow interrupt enable.	0x0
		0	FIFO 0 overflow interrupts are not enabled.	
		1	FIFO 0 overflow interrupts are enabled.	
2	FWMIE1		FIFO1 watermark interrupt enable.	0x0
		0	FIFO1 watermark interrupts are not enabled.	
		1	FIFO1 watermark interrupts are enabled.	
3	FOFIE1		Result FIFO1 overflow interrupt enable	0x0
		0	No result FIFO1 overflow has occurred since the last time the flag was cleared.	
		1	At least one result FIFO1 overflow has occurred since the last time the flag was cleared.	
7:4	-		Reserved.	0x0
8	TEXC_IE		Trigger exception interrupt enable.	0x0
		0	Trigger exception interrupts are disabled.	
		1	Trigger exception interrupts are enabled.	
15:9	-		Reserved.	0x0
31:16	TCOMP_IE		Trigger completion interrupt enable.	0x0
		0	Trigger completion interrupts are disabled.	
		1	Trigger completion interrupts are enabled for trigger source 0 only.	
		2	Trigger completion interrupts are enabled for trigger source 1 only.	
		0b0000000000000011-0b1111111111111110	Associated trigger completion interrupts are enabled.	
		65535	Trigger completion interrupts are enabled for all trigger sources.	

39.6.6 DMA enable register

This register allows to enable or disable the DMA operation of the ADC. DMA operations with ADC channels require special attention. When DMA operations are enabled in ADC registers (FWMDE0 = 1 in DE register, and/or FWMDE =1 in DE register), the ADC FIFO watermark level must be set to a minimum value of 2. See ADC registers (FWMARK bit in FCTRL0 register, and FWMARK bit in FCTRL1)

Table 734. DMA enable register (DE, offset 0x1C)

Bit	Symbol	Value	Description	Reset value
0	FWMDE0		FIFO 0 watermark DMA enable.	0x0
		0	DMA request disabled.	
		1	DMA request enabled.	
1	FWMDE1		FIFO1 watermark DMA enable.	0x0
		0	DMA request disabled.	
		1	DMA request enabled.	
31:2	-		Reserved.	0x0

39.6.7 ADC configuration register

The configuration register controls ADC functions that are common to all commands. The CFG cannot be changed while the CTRL[ADCEN] bit is set. Writes to CFG while ADCEN is set are ignored

Table 735. ADC configuration register (CFG, offset = 0x20)

Bit	Symbol		Description	Reset value
1:0	TPRICTRL		ADC trigger priority control.	0x0
		0	If a higher priority trigger is detected during command processing, the current conversion is aborted and the new command specified by the trigger is started.	
		1	If a higher priority trigger is received during command processing, the current command is stopped after completing the current conversion. If averaging is enabled, the averaging loop will be completed. However, CMDHa[LOOP] is ignored and the higher priority trigger is serviced.	
		2	If a higher priority trigger is received during command processing, the current command will be completed (averaging, looping, compare) before servicing the higher priority trigger.	
		3	Reserved.	
3:2	-		Reserved.	0x0
5:4	PWRSEL		Power configuration select	0x0
		0	Lowest power setting.	
		1	Reserved.	
		2	Reserved.	
		3	Reserved.	
7:6	REFSEL		Voltage reference selection	0x2
		0	Reserved.	
		1	VREFH = voltage on VDDA pin.	
		2	VREFH = voltage on VREFP pin.	
		3	Reserved.	
8	TRES		Trigger resume enable.	0x0
		0	Trigger sequences interrupted by a high priority trigger exception will not be automatically resumed or restarted.	
		1	Trigger sequences interrupted by a high priority trigger exception will be automatically resumed or restarted.	
9	TCMDRES		Trigger command resume.	0x0
		0	Trigger sequences interrupted by a high priority trigger exception will be automatically restarted.	
		1	Trigger sequences interrupted by a high priority trigger exception will be resumed from the command executing before the exception.	
10	HPT_EXDI		High priority trigger exception disable.	0x0
		0	High priority trigger exceptions are enabled.	
		1	High priority trigger exceptions are disabled.	
14:11	-	-	Reserved.	0x0
15	-	-	Reserved.	0x0
23:16	PUDLY		Power up delay.	0x80

Table 735. ADC configuration register (CFG, offset = 0x20) *…continued*

Bit	Symbol		Description	Reset value
27:24	-	-	Reserved.	0x0
28	PWREN		ADC analog pre-enable.	0x0
		0	ADC analog circuits are only enabled while conversions are active. Performance is affected due to analog startup delays.	
		1	ADC analog circuits are pre-enabled and ready to execute conversions without startup delays (at the cost of higher DC current consumption). When PWREN is set, the power up delay is enforced so that any detected trigger does not begin ADC operation until the power up delay time has passed.	
31:29	-	-	Reserved.	0x0

39.6.8 ADC pause register

The Pause register controls an optional inserted delay between conversions.

Note: the PAUSE register should not be modified while the CTRL[ADCEN] bit is set.

Table 736. ADC pause register (PAUSE, offset = 0x24)

Bit	Symbol	Value	Description	Reset value
8:0	PAUSEDLY		Pause delay.	0x0
30:9	-		Reserved.	0x0
31	PAUSEEN		PAUSE option enable.	0x0
		0	Pause operation disabled.	
		1	Pause operation enabled.	

39.6.9 Software trigger register

The Software Trigger Register (SWTRIG) is written to initiate software triggered conversions. Writes to SWTRIG register are ignored while CTRL[ADCEN] is clear.

Note: There is an approximately 3 ADC Clock cycle synchronization delay between asserting ADCEN until SWTRIG can be accepted.

Table 737. Software trigger register (SWTRIG, offset 0x34)

Bit	Symbol	Value	Description	Reset value
0	SWT0		Software trigger 0 event.	0x0
		0	No trigger 0 event generated.	
		1	Trigger 0 event generated.	
1	SWT1		Software trigger 1 event.	0x0
		0	No trigger 1 event generated.	
		1	Trigger 1 event generated.	
2	SWT2		Software trigger 2 event.	0x0
		0	No trigger 2 event generated.	
		1	Trigger 2 event generated.	
3	SWT3		Software trigger 3 event.	0x0
		0	No trigger 3 event generated.	
		1	Trigger 3 event generated.	
4	SWT4		Software trigger 4 event.	0x0
		0	No trigger 4 event generated.	
		1	Trigger 4 event generated.	
5	SWT5		Software trigger 5 event.	0x0
		0	No trigger 5 event generated.	
		1	Trigger 5 event generated.	
6	SWT6		Software trigger 6 event.	0x0
		0	No trigger 6 event generated.	
		1	Trigger 6 event generated.	
7	SWT7		Software trigger 7 event.	0x0
		0	No trigger 7 event generated.	
		1	Trigger 7 event generated.	
8	SWT8		Software trigger 8 event.	0x0
		0	No trigger 8 event generated.	
		1	Trigger 8 event generated.	
9	SWT9		Software trigger 9 event.	0x0
		0	No trigger 9 event generated.	
		1	Trigger 9 event generated.	
10	SWT10		Software trigger 10 event.	0x0
		0	No trigger 10 event generated.	
		1	Trigger 10 event generated.	

Table 737. Software trigger register (SWTRIG, offset 0x34) *…continued*

Bit	Symbol	Value	Description	Reset value
11	SWT11		Software trigger 11 event.	0x0
		0	No trigger 11 event generated.	
		1	Trigger 11 event generated.	
12	SWT12		Software trigger 12 event.	0x0
		0	No trigger 12 event generated.	
		1	Trigger 12 event generated.	
13	SWT13		Software trigger 13 event.	0x0
		0	No trigger 13 event generated.	
		1	Trigger 13 event generated.	
14	SWT14		Software trigger 14 event.	0x0
		0	No trigger 14 event generated.	
		1	Trigger 14 event generated.	
15	SWT15		Software trigger 15 event.	0x0
		0	No trigger 15 event generated.	
		1	Trigger 15 event generated.	
31:16	-		Reserved.	0x0

39.6.10 Trigger status register

This register contains status flags to indicate when trigger sequences have been completed or interrupted by a high priority trigger exception. Each bit in this register is set by hardware and cleared by software.

To clear a bit in this register, write a 0b1 to the corresponding bit position.

Table 738. Trigger status register (TSTAT, offset 0x38)

Bit	Symbol	Value	Description	Reset value
15:0	TEXC_NUM		Trigger exception number.	0x0
		0	No triggers have been interrupted by a high priority exception. Or CFG[TRES] = 1.	
		1	Trigger 0 is interrupted by a high priority exception.	
		2	Trigger 1 is interrupted by a high priority exception.	
		0b0000000000000011-0b1111111111111110	Associated trigger sequence is interrupted by a high priority exception.	
		65535	Every trigger sequence is interrupted by a high priority exception.	

Table 738. Trigger status register (TSTAT, offset 0x38) *…continued*

Bit	Symbol	Value	Description	Reset value
31:16	TCOMP_FLAG		Trigger completion flag.	0x0
		0	No triggers are completed. Trigger completion interrupts are disabled.	
		1	Trigger 0 is completed and trigger 0 has enabled completion interrupts.	
		2	Trigger 1 is completed and trigger 1 has enabled completion interrupts.	
		0b0000000000000011-0b1111111111111110	Associated trigger sequence is completed and has enabled completion interrupts.	
		65535	Every trigger sequence is completed and every trigger has enabled completion interrupts.	

39.6.11 ADC offset trim register

The ADC offset trim register is used to trim for offset.
The ADC supports a calibration step where the ADC is configured to perform a calibration operation to determine the value needed in the OFSTRIM register. Set the CALOFS bit to determine the value to put in the OFSTRIM register, This automatically begins a sequence to calculate the value.

Once the sequence has completed, the OFSTRIM register is updated with a signed value between -16 and 15. This value is used to minimize offset during normal operation.

Table 739. ADC offset trim register (OFSTRIM, offset = 0x40)

Bit	Symbol	Description	Reset value
4:0	OFSTRIM_A	Trim for offset.	0x0
15:5	-	Reserved.	0x0
20:16	OFSTRIM_B	Trim for offset.	0x0
31:21	-	Reserved.	0x0

39.6.12 Trigger control registers

The Trigger Control ($TCTRL_a$) registers implement control fields associated with each implemented trigger source. When the ADC is actively executing commands, only one of the $TCTRL_a$ registers is actively controlling ADC conversions. The actively controlling $TCTRL_a$ register must not be updated while the ADC is active. A write to a $TCTRL_a$ register while that trigger control register is controlling ADC operation might result in unpredictable behavior.

Table 740. Trigger control registers (TCTRL[0:15], offsets 0xA0 to 0xDC)

Bit	Symbol	Value	Description	Reset value
0	HTEN		Trigger enable.	0x0
		0	Hardware trigger source disabled.	
		1	Hardware trigger source enabled.	
1	FIFO_SEL_A		SAR result destination for SE mode, or channel A if DSE mode is used.	0x0
		0	Result written to FIFO 0.	
		1	Result written to FIFO 1.	
2	FIFO_SEL_B		SAR result destination for channel B if DSE mode is used.	0x0
		0	Result written to FIFO 0.	
		1	Result written to FIFO 1.	
7:3			Reserved.	0x0
11:8	TPRI		Trigger priority setting.	0x0
		0	Set to highest priority, Level 1.	
		0b0001-0b1110	Set to corresponding priority level.	
		15	Set to lowest priority, Level 16.	
14:12			Reserved.	0x0
15	RSYNC		Trigger resync.	0x0
19:16	TDLY		Trigger delay select.	0x0

Table 740. Trigger control registers (TCTRL[0:15], offsets 0xA0 to 0xDC) *…continued*

Bit	Symbol	Value	Description	Reset value
23:20			Reserved.	0x0
27:24	TCMD		Trigger command select.	0x0
		0	Not a valid selection from the command buffer. Trigger event is ignored.	
		1	CMD1 is executed.	
		0b0010-0b1110	Corresponding CMD is executed.	
		15	CMD15 is executed.	
31:28			Reserved.	0x0

39.6.13 ADC FIFO control registers

The FIFO Control ($FCTRL_a$) registers contain control and status fields for each FIFO in the design.

A programmable watermark can be set for each FIFO that can be used to trigger an interrupt. In addition, the number of entries stored in each FIFO can be monitored by reading FCTRLa[FCOUNT]. DMA operations with ADC channels require special attention. When DMA operations are enabled in ADC registers (FWMDE0 = 1 in DE register, and/or FWMDE =1 in DE register), the ADC FIFO watermark level must be set to a minimum value of 2. See ADC registers (FWMARK bit in FCTRL0 register, and FWMARK bit in FCTRL1)

Table 741. ADC FIFO control registers (FCTRL[0:1], offsets 0xE0 to 0xE4)

Bit	Symbol	Value	Description	Reset value
4:0	FCOUNT		Result FIFO counter.	0x0
15:5	-		Reserved.	0x0
19:16	FWMARK		Watermark level selection. When DMA operations are enabled in ADC registers (FWMDE0 = 1 in DE register, and/or FWMDE = 1 in DE register), the ADC FIFO watermark level must be set to a minimum value of 2.	0x0
31:20	-		Reserved.	0x0

39.6.14 Gain calibration control registers

The Gain Calibration Control (GCCa) registers are utilized as part of the auto-calibration routine. The GAIN_CAL field of this register is calculated during auto-calibration and stored in the GCCa register for user calculations. There is a RDY status flag in the GCC register that indicates whether the value GCCa[GAIN_CAL] is valid.

After the auto-calibration sequence has calculated the correct GCCa[GAIN_CAL] the GCCa[RDY] bit is asserted automatically.

Note: Requesting an auto-calibration will automatically clear the GCCa[RDY] bit until the GCCa[GAIN_CAL] value is calculated.

To complete the auto-calibration routine, the GCCa[GAIN_CAL] must be utilized to calculate the gain calculation result.

For more information see Section 39.7.5.3 “Calibration”.

The register field GAIN_CAL holds a 16-bit number with 15-bits representing whole numbers and 1 bit (Bit 0) fractional.

Table 742. Gain calibration control registers (GCC[0:1], offsets 0xF0 to 0xF4)

Bit	Symbol	Value	Description	Reset value
15:0	GAIN_CAL		Gain calibration value.	0x0
23:16			Reserved.	0x0
24	RDY		Gain calibration value valid.	0x0
		0	The gain calibration value is invalid. Run the auto-calibration routine for this value to be written.	
		1	The gain calibration value is valid. It should be used to update the GCRa[GCALR] register field.	
31:25	-		Reserved.	0x0

39.6.15 Gain calculation result registers

The Gain Calculation Result (GCR_a) registers are utilized as part of the auto-calibration routine.

There is a RDY status flag in the GCR register which indicates whether the value GCRa[GCALR] is valid. After writing the GCRa[GCALR] value, assert GCRa[RDY] to indicate that the offset calculation result is valid.

After beginning a calibration sequence by asserting CTRL[CAL_REQ], the calibration sequence is not completed until GCRa[GCALR] is calculated and GCRa[RDY] is asserted.

The gain calculation results in a floating-point value between 1 and 2. This value is always between 1 and 2, therefore, the leading MSB 1 is redundant and does not have to be stored in this register. GCRa[GCALR] should hold the 16-bit fractional component of the gain offset calculation. In other words, the value to store in GCRa[GCALR] = gain_calculation - 1.

To convert the GCRa[GCALR] value back into decimal format, this would be calculated as: 1+0.5*GCRa[15]+0.25*GCRa[14]+0.125*GCRa[13]+...

For more information see Section 39.7.5.3 “Calibration”.

Table 743. Gain calculation result (GCR[0:1], offsets 0xF8 to 0xFC)

Bit	Symbol	Value	Description	Reset value
15:0	GCALR		Gain calculation result.	0x0
23:16	-		Reserved.	0x0
24	RDY		Gain calculation ready.	0x0
		0	The gain offset calculation value is invalid.	
		1	The gain calibration value is valid.	
31:25	-		Reserved.	0x0

39.6.16 ADC command low buffer registers

There are 15 command buffers (CMD_a), each constructed from two 32-bit registers ($CMDL_a$:$CMDH_a$) that can be configured for different channel select and varying conversion options. Any of the command buffers is selected and used as the controlling command by association to a trigger event via configuration of the $TCTRL_a$[TCMD] bit field. When the ADC is actively executing commands, only one of the CMD buffers is actively controlling ADC conversions. The actively controlling CMD buffer must not be updated while the ADC is active. A write to a CMD buffer while that CMD buffer is controlling ADC operation might result in unpredictable behavior.

Table 744. ADC command low buffer registers (CMDL[1:15], offsets 0x100 to 0x170)

Bit	Symbol	Value	Description	Reset value
4:0	ADCH		Input channel select	0x0
		0	Select CH0A or CH0B or CH0A/CH0B pair.	
		1	Select CH1A or CH1B or CH1A/CH1B pair.	
		2	Select CH2A or CH2B or CH2A/CH2B pair.	
		3	Select CH3A or CH3B or CH3A/CH3B pair.	
		0b00100-0b11101	Select corresponding channel CHnA or CHnB or CHnA/CHnB pair.	
		30	Select CH30A or CH30B or CH30A/CH30B pair.	
		31	Select CH31A or CH31B or CH31A/CH31B pair.	
6:5	CTYPE		Conversion type.	0x0
		0	Single-ended mode. Only A side channel is converted.	
		1	Single-ended mode. Only B side channel is converted.	
		2	Differential mode. A-B.	
		3	Dual-Single-Ended Mode. Both A side and B side channels are converted independently but both side need to be start conversion at same time. if not, the SideA and SideB cannot convert independently.	
7	MODE		Select resolution of conversions	0x0
		0	Standard resolution. Single-ended 12-bit conversion, Differential 13-bit conversion with 2’s complement output.	
		1	High resolution. Single-ended 16-bit conversion; Differential 16-bit conversion with 2’s complement output.	
31:8	-		Reserved.	0x0

39.6.17 ADC command high buffer registers

There are 15 command buffers (CMD_a), each constructed from two 32-bit registers ($CMDL_a$:$CMDH_a$) that can be configured for different channel select and varying conversion options. Any of the command buffers is selected and used as the controlling command by association to a trigger event via configuration of the $TCTRL_a$[TCMD] bit field. When the ADC is actively executing commands, only one of the CMD buffers is actively controlling ADC conversions. The actively controlling CMD buffer must not be updated while the ADC is active. A write to a CMD buffer while that CMD buffer is controlling ADC operation might result in unpredictable behavior.

Table 745. ADC command high buffer registers (CMDH[1:15], offsets 0x104 to 0x174))

Bit	Symbol	Value	Description	Reset value
1:0	CMPEN		After an ADC channel input has been processed, the CMDHa (CMPEN) field directs the compare function to optionally store the value when the compare operation is true. When compare is enabled, the conversion result is compared with the compare value registers (CVa[CVH] and CVa[CVL]). There are multiple options on command sequencing related to the compare function as described in Section 39.6.18 “Compare value registers”. Note that not all Command Buffers have implemented the CMPEN field which is only available in Command Buffers that have a corresponding Compare Value register (i.e., CMDH 1-4).	0x0
		00b	Compare disabled.	
		01b	Reserved.	
		10b	Compare enabled. Store on true.	
		11b	Compare enabled. Repeat channel acquisition (sample/convert/compare) until true.	
2	WAIT_TRIG		Wait for trigger assertion before execution.	0x0
		0	This command will be automatically executed.	
		1	The active trigger must be asserted again before executing this command.	
6:3			Reserved.	0x0
7	LWI		Loop with increment.	0x0
		0	Auto channel increment disabled.	
		1	Auto channel increment enabled.	

Table 745. ADC command high buffer registers (CMDH[1:15], offsets 0x104 to 0x174) *…continued*

Bit	Symbol	Value	Description	Reset value
10:8	STS		Sample time select. When programmed to 000 the minimum sample time of 3.5 ADCK cycles is selected. When STS is programmed to a non-zero value the sample time is (3.5 + 2^STS) ADCK cycles. The shortest sample time maximizes conversion speed for lower impedance inputs. Extending sample time allows higher impedance inputs to be accurately sampled. Longer sample times can also be used to lower overall power consumption when command looping and sequencing is configured and high conversion rates are not required.	0x0
		0	Minimum sample time of 3.5 ADCK cycles.	
		1	3.5 + 2^1 ADCK cycles; 5.5 ADCK cycles total sample time.	
		2	3.5 + 2^2 ADCK cycles; 7.5 ADCK cycles total sample time.	
		3	3.5 + 2^3 ADCK cycles; 11.5 ADCK cycles total sample time.	
		4	3.5 + 2^4 ADCK cycles; 19.5 ADCK cycles total sample time.	
		5	3.5 + 2^5 ADCK cycles; 35.5 ADCK cycles total sample time.	
		6	3.5 + 2^6 ADCK cycles; 67.5 ADCK cycles total sample time.	
		7	3.5 + 2^7 ADCK cycles; 131.5 ADCK cycles total sample time.	
11	-		Reserved.	0x0
14:12	AVGS		Hardware average select.	0x0
		0	Single conversion.	
		1	2 conversions averaged.	
		2	4 conversions averaged.	
		3	8 conversions averaged.	
		4	16 conversions averaged.	
		5	32 conversions averaged.	
		6	64 conversions averaged.	
		7	128 conversions averaged.	
15	-		Reserved.	0x0
19:16	LOOP		Loop Count Select.	0x0
		0	Looping not enabled. Command executes 1 time.	
		1	Loop 1 time. Command executes 2 times.	
		2	Loop 2 times. Command executes 3 times.	
		0b0011-0b1110	Loop corresponding number of times. Command executes LOOP+1 times.	
		15	Loop 15 times. Command executes 16 times.	
23:20	-		Reserved.	0x0
27:24	NEXT		Next Command Select	0x0
		0	No next command defined. Terminate conversions at completion of current command. If lower priority trigger pending, begin command associated with lower priority trigger.	
		1	Select CMD1 command buffer register as next command.	
		0b0010-0b1110	Select corresponding CMD command buffer register as next command	
		15	Select CMD15 command buffer register as next command.	
31:28	-		Reserved.	0x0

39.6.18 Compare value registers

The compare value registers (CV_a) contain values used to compare the conversion result when the compare function is enabled. This register is formatted in the same way as the D field in the RESFIFO registers in different modes of operation for both bit position definition and value format using unsigned or signed 2's complement. There is a direct association of each compare value register to a specific command buffer register (that is, CV1 is only used during execution of CMD1 command).

When the ADC is actively executing commands, the CV_a register associated with the active command (CMD_a) must not be updated. Writes to associated CV_a register during this time might result in unpredictable behavior.

Table 746. Compare value register (CV[1:4], offsets 0x200 to 0x20C)

Bit	Symbol	Value	Description	Reset value
15:0	CVL		Compare value low.	0x0
31:16	CVH		Compare value high.	0x0

39.6.19 ADC data result FIFO register0

The data result FIFO register (RESFIFO) is a 16 entry FIFO that stores the data result of ADC conversions. In addition, several tag fields of source command and trigger information are stored along with the data. FCTRL[FCOUNT] indicates how many valid data words are stored in the RESFIFO. Reading RESFIFO provides the oldest unread data word entry in the FIFO and decrements FCOUNT. The FIFO can be emptied by successive reads of RESFIFO. The FIFO is reset by writing 0b1 to the CTRL[RSTFIFO] bit.

Table 747 describes the format of data in the result FIFO in the different modes of operation. The sign bit is the (MSB) in signed 2's complement modes. For example, when configured for 12-bit single-ended mode, D[15] and D[2:0] are cleared. When configured for 13-bit differential mode, D[15] is the sign bit and D[2:0] are cleared.

Table 747. Data result register format description

Conversion mode	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Format
16-bit differential	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	Signed 2’s complement
16-bit single ended	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	Unsigned, 16-bit magnitude
13-bit differential	S	D	D	D	D	D	D	D	D	D	D	D	D	0	0	0	Signed 2’s complement
12-bit single ended	0	D	D	D	D	D	D	D	D	D	D	D	D	0	0	0	Unsigned, zero in D[15] and D[2:0]

Table 748. ADC Data Result FIFO Register (RESFIFO0, offset 0x300)

Bit	Symbol	Value	Description	Reset value
15:0	D		Data result.	0x0

Table 748. ADC Data Result FIFO Register (RESFIFO0, offset 0x300) *…continued*

Bit	Symbol	Value	Description	Reset value
19:16	TSRC		Trigger source.	0x0
		0	Trigger source 0 initiated this conversion.	
		1	Trigger source 1 initiated this conversion.	
		0b0010-0b1110	Corresponding trigger source initiated this conversion.	
		15	Trigger source 15 initiated this conversion.	
23:20	LOOPCNT		Loop count value.	0x0
		0x0	Result is from initial conversion in command.	
		0x1	Result is from second conversion in command.	
		0x2 - 0xE	Result is from LOOPCNT+1 conversion in command.	
		0xF	Result is from 16th conversion in command.	
27:24	CMDSRC		Command buffer source.	0x0
		0	Not a valid CMDSRC value for a data word in RESFIFO. 0x0 is only found in initial FIFO state prior to an ADC conversion.	
		1	CMD1 buffer used as control settings for this conversion.	
		0b0010-0b1110	Corresponding command buffer used as control settings for this conversion.	
		15	CMD15 buffer used as control settings for this conversion.	
30:28	-		Reserved.	0x0
31	VALID		FIFO entry is valid	0x0
		0	FIFO is empty. Discard any read from RESFIFO.	
		1	FIFO record read from RESFIFO is valid.	

39.6.20 ADC data result FIFO register1

The data result FIFO register (RESFIFO) is a 16 entry FIFO that stores the data result of ADC conversions. In addition, several tag fields of source command and trigger information are stored along with the data. FCTRL[FCOUNT] indicates how many valid data words are stored in the RESFIFO. Reading RESFIFO provides the oldest unread data word entry in the FIFO and decrements FCOUNT. The FIFO can be emptied by successive reads of RESFIFO. The FIFO is reset by writing 0b1 to the CTRL[RSTFIFO] bit.

Table 749. ADC Data Result FIFO Register (RESFIFO1, offset = 0x304)

Bit	Symbol	Value	Description	Reset value
15:0	D		Data result.	0x0
19:16	TSRC		Trigger source.	0x0
		0	Trigger source 0 initiated this conversion.	
		1	Trigger source 1 initiated this conversion.	
		0b0010-0b1110	Corresponding trigger source initiated this conversion.	
		15	Trigger source 15 initiated this conversion.	

Table 749. ADC Data Result FIFO Register (RESFIFO1, offset = 0x304) *…continued*

Bit	Symbol	Value	Description	Reset value
23:20	LOOPCNT		Loop count value.	0x0
		0x0	Result is from initial conversion in command.	
		0x1	Result is from second conversion in command.	
		0x2 - 0xE	Result is from LOOPCNT+1 conversion in command.	
		0xF	Result is from 16th conversion in command.	
27:24	CMDSRC		Command buffer source.	0x0
		0	Not a valid value CMDSRC value for a data word in RESFIFO. 0x0 is only found in initial FIFO state prior to an ADC conversion.	
		1	CMD1 buffer used as control settings for this conversion.	
		0b0010-0b1110	Corresponding command buffer used as control settings for this conversion.	
		15	CMD15 buffer used as control settings for this conversion.	
30:28			Reserved.	0x0
31	VALID		FIFO entry is valid.	0x0
		0	FIFO is empty. Discard any read from RESFIFO.	
		1	FIFO record read from RESFIFO is valid.	

39.6.21 Calibration general A-side registers

The A-side general calibration value registers contain calibration information that is generated by the calibration function.

CAL_GAR registers are automatically set once the self calibration sequence is done (STAT[CAL_RDY] is set). If these registers are modified after calibration, the linearity error specifications may not be met. The calibration values CAL_GAR will affect the end conversion result by conditionally subtracting their values from the conversion before end result is transferred into the FIFOs. Calibration must be run each time the ADC is powered down or a hard reset is issued.

To reduce the latency required to run calibration, the CAL_GAR values can be stored in non-volatile memory after an initial calibration and recovered prior to the first ADC conversion. These values are only read write accessible when the ADC is disabled with CTRL[ADCEN] = 0b0.

Note: The access time when writing to these registers will be larger than 3 ADC clock cycles. Wait states will be inserted on the bus to meet synchronization timing to the associated CAL_GAR register. For more information, see Section 39.7.5.3 “Calibration”.

Note, the width of each register in this array is non-uniform. The exact width of each register is defined in Section 39.7.10 “Calibration general A-side and B-side widths”.

Table 750. Calibration general A-side registers (CAL_GAR[0:32], offsets 0x400 to 0x480)

Bit	Symbol	Value	Description	Reset value
15:0	CAL_GAR_VAL		Calibration general A side register element.	0x0
31:16	-		Reserved.	0x0

39.6.22 Calibration general B-side registers

The B-side general calibration value registers contain calibration information that is generated by the calibration function.

CAL_GBR registers are automatically set once the self calibration sequence is done (STAT[CAL_RDY] is set). If these registers are modified after calibration, the linearity error specifications may not be met. These calibration values CAL_GBR will affect the end conversion result by conditionally subtracting their values from the conversion before end result is transferred into the FIFOs. Calibration must be run each time the ADC is powered down or a hard reset is issued.

To reduce the latency required to run calibration, the CAL_GBR values can be stored in non-volatile memory after an initial calibration and recovered prior to the first ADC conversion. These values are only read write accessible when the ADC is disabled with CTRL[ADCEN] = 0b0.

Note: The access time when writing to these registers will be larger than 3 ADC clock cycles. Wait states will be inserted on the bus to meet synchronization timing to the associated CAL_GBR register. The width of each register in this array is non-uniform.

For more information, see Section 39.7.5.3 “Calibration”.

Note, the width of each register in this array is non-uniform. The exact width of each register is defined in Section 39.7.10 “Calibration general A-side and B-side widths”.

Table 751. Calibration general B-side registers (CAL_GBR[0:32], offsets 0x500 to 0x580)

Bit	Symbol	Value	Description	Reset value
15:0	CAL_GBR_VAL		Calibration general B side register element.	0x0
31:16	-		Reserved.	0x0

39.7 Functional description

39.7.1 Command sequencing

The ADC performs analog-to-digital conversions on any of the software selectable analog input channels by a successive approximation algorithm. The ADC module initializes to its lowest power state during reset and in Low-Power Stop mode. The ADC analog circuits can optionally be pre-enabled for faster starts to conversions at the expense of higher idle currents. Conversions are initiated by selectable trigger events from software or hardware sources. The trigger detect logic includes a configurable enable and priority scheme for the available trigger sources. The ADC includes multiple command buffers that provide configurable flexibility for channel scanning and independent channel selections for different trigger sources. Multiple command buffers also allow variable option selection such as differential vs. single-ended sample time and averaging on a per-channel basis.

The ADC module optionally averages the result of multiple conversions on a channel before storing the calculated result. The hardware average function is enabled by setting CMDHa[AVGS] bit field to a non-zero value and operates in any of the conversion modes and configurations.

When the conversion and averaging loops are completed, the resulting data is placed in one of 2 available FIFO data buffers along with other tag information associated with the result. A configurable watermark level supports interrupt or DMA requests when the number of stored data words exceeds the setting. Interrupts can also be enabled to indicate when FIFO overflow errors occur.

The ADC module optionally compares the result of a conversion with the contents of two value registers for less-than, greater-than, inside-range or outside-range detection. The compare function operates in any of the conversion modes and configurations.

The ADC module includes offset and linearity calibration logic. A request for calibration should be made any time upon reset or power up. Each SAR conversion will utilize calibration data calculated during the auto-calibration routine.

Figure 148 shows the sequencing of a ADC command.

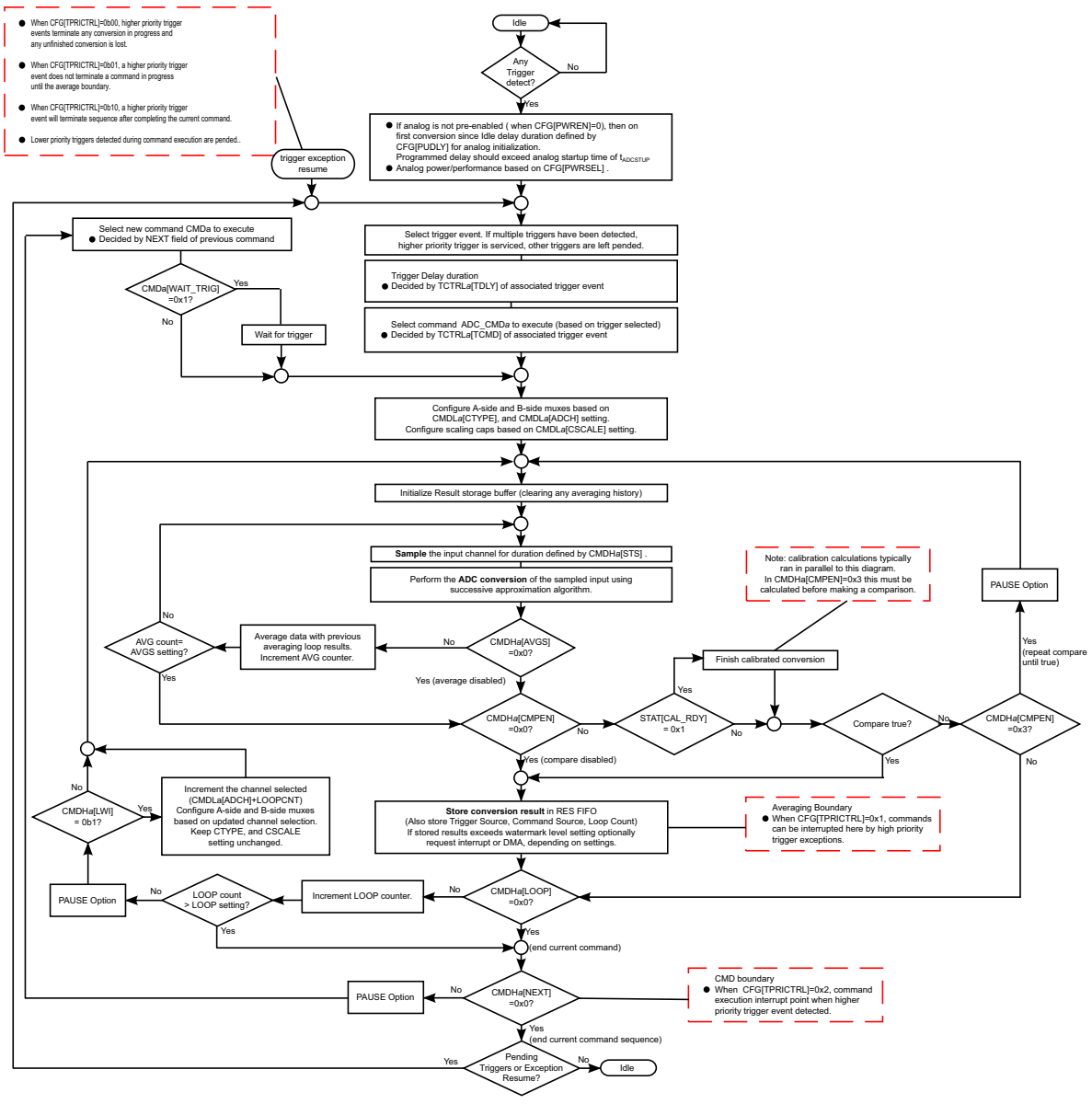


Fig 148. ADC command sequencing

39.7.1.1 ADC start-up sequence software work-around

The ADC must be started up in a state with PWREN = 0x0 and PUDLY > 0x0.

```
// disable for init
adc0.adc_disable();
adc0.adc_pwr_disable();

// Keep PWREN = 0x0 and PUDLY > 0x0, then kick off ADC_EN, then kick off the conversion
     trigger
adc0.adc_enable();
adc0.adc_start_conv();
```

```
//Set PWREN as desired for the application
adc0.adc_pwr_enable();
...
...
//Repeat from the top when disabling the ADC with ADC_EN = 0x0
//When shutting off the ADC, you must also set PWREN = 0x0
```

39.7.2 Voltage reference

The voltage reference high (VREFH) used by the ADC is supplied from an off-chip source supplied through the VREFP or VDDA pins. VREFL is always from an external pin and must be at the same voltage potential as VSSA.

This instance of the ADC block supports a programmable selection of the Voltage Reference High used for ADC conversions (via the CFG[REFSEL] field). See Section 39.4.1 “ADC signal descriptions”.

39.7.3 Power control

The default setting for the ADC analog circuits is disabled while the ADC is in its Idle state. When a trigger is detected and ADC command processing is initiated, the analog circuits are enabled and require a period of initialization before the first conversion cycle. The CFG[PUDLY] should be programmed such that a delay duration longer than tADCSTUP is incurred. Accuracy of the initial conversion(s) after activation is degraded if CFG[PUDLY] is set to too small a value.

Set PUDLY so that the analog startup time is greater than tADCSTUP,

PUDLY * 4 * ADCclock > tADCSTUP (1)

For instance, for a 5 microsecond startup time with a 24 MHz ADC clock, set PUDLY to 0x1E.

Faster conversion startup times can be achieved by optionally setting the CFG[PWREN] field to pre-enable the analog circuits of the ADC at the expense of power consumption even while the ADC is in an idle state. When PWREN is set, the Power Enable timer is activated and enforces the minimum startup (controlled by the PUDLY register field) before detected triggers are allowed to initiate ADC conversions.

The ADC also has power option settings for controlling power and performance. See Table 752.

Table 752. Power option settings

CFG[PWRSEL]	Description
0b00	Slowest speed/lowest power setting. Maximum ADC input clock frequency = 24 MHz.

39.7.4 Clock operation

The ADC operates from the ADCK clock input provided from an on-chip clock select block and is used by the SAR conversion control sequencing logic and the FIFO storage buffer. The ADCK frequency must fall within the specified frequency range for ADCK and will vary based on configuration of CFG[PWRSEL]. The clock sources for ADC are:

- MAIN_CLK
- PLL0
- FRO_HF

The ADC target sampling rate is 1 mega sample per second. See Figure 4.

The ADC continues operating in stop and wait modes provided the Doze Enable bit (CTRL[DOZEN]) is clear and the on-chip clock select block continues to supply an ADCK clock source.

Remark: In stop mode with CTRL[DOZEN] == 0b0, the bus clock will be shut off, and asynchronous interrupts and DMA requests can be configured.

In addition, the ADC continues to process commands and write data to the internal FIFO.

The ADC has four sources for asynchronous interrupts during stop mode:

- Watermark
- FIFO overflow
- TCOMP
- TEXC

To enable them, properly configure the bits IE[FWMIEx], IE[FOFIEx], IE[TCOMP_IE], and IE[TEXC_IE] before entering stop mode.

When the DOZEN bit is set in stop and wait modes the ADC will wait for the current averaging iteration/FIFO storage to complete before acknowledging stop or wait mode entry. Any pending triggers will be dropped when a stop/wait mode request is made with DOZEN set.

The ADC is forced into its lowest power setting after acknowledging the DOZEN stop/wait mode request. The same behavior will be observed when entering a Low Leakage Stop Mode

39.7.5 Trigger detect and command execution

ADC command execution is initiated from up to 16 trigger sources. Each trigger can be software generated by writing 0b1 to the corresponding SWTRIG[SWTn] bitfield.

Alternatively, hardware triggers can be generated from asynchronous input sources at the periphery of the ADC.

Table 753. ADC hardware triggers

Hardware trigger	Mapped to
0	GPIO irq_pint[0]
1	GPIO irq_pint[1]
2	State Configurable Timer (SCT) sct0_outputs[4]
3	State Configurable Timer (SCT) sct0_outputs[5]
4	State Configurable Timer (SCT) sct0_outputs[9]
5	State Counter Timer (CTIMER) ct0_mat3_out
6	State Counter Timer (CTIMER) ct1_mat3_out

Table 753. ADC hardware triggers *...continued*

Hardware trigger	Mapped to
7	State Counter Timer (CTIMER) ct2_mat3_out
8	State Counter Timer (CTIMER) ct3_mat3_out
9	State Counter Timer (CTIMER) ct4_mat3_out
10	Comparator
11	ARM tx event
12	GPIO BMATCH

Each hardware trigger source is enabled by setting the associated enable bit (TCTRLa[HTEN]).

Each trigger source is assigned a priority via the associated priority control field (TCTRLa[TPRI]). Each of the trigger sources is associated with a command buffer via the associated command select field (TCTRLa[TCMD]).

When a hardware trigger input is enabled, hardware trigger events are detected on the rising-edge of the associated hardware trigger source. The hardware trigger event must be high for 1.5 ADCK cycles.

Each trigger source has an associated priority field TCTRLa[TPRI] that allows for arbitration between trigger sources.

Arbitration is in control of selecting which trigger sequence to execute next, and selecting how to handle a trigger exception. Trigger exceptions are defined as allowing a higher priority trigger sequence to interrupt operation of a lower priority sequence.

When a trigger exception occurs, programmable arbitration allows the configurable stop and resume points for low priority sequences. The fields affecting arbitration are CFG[HPT_EXDI], CFG[TCMDRES], CFG[TRES] and CFG[TPRICTRL].

- If CFG[HPT_EXDI] is set to 1'b1 then trigger exceptions are disabled and any higher priority triggers will be left pending until the current sequence completes. Note that new triggers are accepted based on priority.
- If CFG[HPT_EXDI] is set to 1'b0 (default), then exceptions are enabled and the higher priority sequence will begin executing at a user specified breakpoint. Breakpoint locations are determined by the register CFG[TPRICTRL]. CFG[TPRICTRL] will have an effect on latency for accepting a trigger exception.
- When TPRICTRL=0x0, a higher priority trigger causes an immediate command abort and the new command specified by the trigger is immediately started.
- When TPRICTRL=0x1, the current conversion is allowed to complete (including averaging) before the higher priority exception is initiated. In this mode, if the command is running through a series of averages, this series is completed. However, there is no requirement to finish the entire command before being interrupted. For example, if the command consists of four loop iterations, there is no requirement to complete all 4 iterations before the interrupt occurs.
- When TPRICTRL=0x2, a higher priority trigger will begin once the current command is completed. If a command consists of 5 loop iterations each containing 8 averages, then all 5x8 conversions must be completed before accepting the trigger exception. CFG[TCMDRES] and CFG[TRES] determine what the ADC will do after accepting a trigger exception.

- If CFG[TRES] = 0x0 then commands will not be automatically resumed after being stopped by an exception. However, an interrupt will be set to indicate this case has occurred. The flag TSTAT[TEXC_NUM] can be to resolve which trigger was stopped by the exception.
- If CFG[TRES] = 0x1 the ADC will automatically resume commands after they were stopped by an exception. By utilizing CFG[TRES] in conjunction with CFG[TCMDRES], the ADC can be programmed to resume commands at one of two possible locations.
- If CFG[TCMDRES] = 0x0 then the trigger which was stopped by an exception will be resumed from the beginning of its associated command sequence. Note, triggers which are waiting to be resumed take the same priority programmed to TCTRLa[TPRI].
- If CFG[TCMDRES] = 0x1 then the trigger will be resumed from the command that it was executing before being interrupted by an exception.

If a lower priority trigger occurs (that is, a trigger event occurs that is configured for a lower priority than the trigger source associated with the currently executing command), the trigger detect is left pending until completion of the current command sequence. Lower priority trigger events cannot be serviced until a higher priority triggered command (or command sequence) completes.

When a conversion is completed (including hardware averaging when AVGS is non-zero), the result is placed in a RESFIFO buffer. When an ADC command selects looping (when LOOP is non-zero) a command will store multiple conversion results to the FIFO during execution of that command.

At the end of command execution, the NEXT field of the command selects the next command to be executed. Multiple commands can be executed sequentially by configuration of each commands NEXT field. Setting the next command to 0x0 causes conversions to terminate at the completion of the command. Unending circular command execution is allowed by setting the NEXT field in the last command in a sequence to the first command in the sequence.

By default, command sequences will execute automatically in the order that NEXT fields are programmed. However, by utilizing the CMDHa[WAIT_TRIG], command execution can be stalled and launched based on trigger inputs. For example, if TRIGGER2 is programmed to start the command sequence CMD1, CMD2, CMD3, then receiving TRIGGER 2 one time will unconditionally run this sequence to completion. If CMDH2[WAIT_TRIG] is set to 0x1, however, then the sequence will pause after CMD1 until TRIGGER2 is received again. Therefore, sequences can be stalled until receiving a trigger assertion.

Disabling the ADC by writing 0b0 to the CTRL[ADCEN] bitfield terminates any active ADC command processing. Writing 0b0 to the ADCEN bitfield causes the current command (or command sequence) to terminate, clears any pending triggers and sends ADC to an IDLE state.

39.7.5.1 Pause option

When the maximum conversion rate is not required by an application the effective conversion rate can be reduced by implementing periodic trigger events to initiate ADC conversions or by selecting a reduced frequency clock as the ADACK source. Both of

these options are dependent on ADC triggering and clocking options external to the ADC block. The latency associated with ADC analog power up delays results in a limit on the maximum conversion rate when using periodic triggering.

Another means of reducing conversion rates is by inserting a pause of a programmable duration between LOOP iterations, between commands in a sequence, and between conversions when command is executing in the "Compare Until True" configuration. When PAUSE[PAUSEEN] is set, the PAUSE[PAUSEDLY] field controls the duration of pausing during command execution sequencing. The pause delay is a count of (PAUSEDLY*4) ADCK cycles. Note, the PAUSE register should not be changed while the CTRL[ADCEN] bit is set. Writes to the PAUSE register while ADCEN is set can lead to metastable operation.

See Figure 148 for the places during command execution sequencing where the pause is optionally inserted.

39.7.5.2 Resync functionality

Any trigger source (SW or HW) can be configured to act as a resync trigger. Trigger based resync functionality is used to interrupt a running trigger (resync target) and clear the FIFO it is writing to. This can either be used to abort a running sequence, or restart a running sequence depending on the configuration of CFG[TRES].

If CFG[TRES] = 0b1 then the target sequence will be aborted, the FIFO cleared, and the sequence will restart after the resync occurs.

If CFG[TRES] = 0b0 then the target sequence will be aborted and the FIFO will be cleared after the RESYNC occurs.

Note: The FIFO(s) cleared are based on the resync target TCTRLm[FIFO_SEL_A] and TCTRLm[FIFO_SEL_B]. To only clear one FIFO, TCTRLm[FIFO_SEL_A] == TCTRLm[FIFO_SEL_B]. Any results not associated with the resync target will be lost if they are stored in either TCTRLm[FIFO_SEL_A] or TCTRLm[FIFO_SEL_B] at the time of the resync.

A resync trigger needs to have a specific target. The resync will only occur if the resync target is running at the time of the trigger.

For the following description, let n be the resync trigger number, let m be the resync target number. According to these variables, trigger n should resync trigger m. To enable a trigger source to act as a resync trigger, the following conditions must be satisfied:

- The resync trigger TCTRLn[RSYNC] must be set to 0b1.
- The resync trigger must have higher priority than the resync target. TCTRLn[TPRI] must be less than TCTRLm[TPRI].
- The resync target is specified using TCTRLn[TCMD]. In this case the resync target, m, must be equal to TCTRLn[TCMD].
- The resync target, m, must be executing commands when the resync trigger, n, is asserted.
- Trigger m must have at least one conversion left to begin when trigger n is received.

If a trigger source n has TCTRLn[RSYNC] set to 0b1, but some of the above conditions aren't met than the trigger source n will be ignored. Figure 149 illustrates a resync trigger sequence executing.

Note: In this example, trigger source 1 is configured to resync trigger source 0.

In Figure 149, trigger source 0 was executing a sequence of commands when trigger source 1 was asserted. The trigger source 0 is stopped when trigger source 1 is asserted (after some synchronization delay). In addition, the FIFO that is written to by the trigger source 0 is cleared (FIFO0 in this example). After the trigger 0 sequence is stopped, the ADC will run the next trigger pending with the highest priority. This is marked as NXT, for next trigger. If resume functionality is enabled, and trigger source 0 had the highest priority pending, then NXT = 0x00.

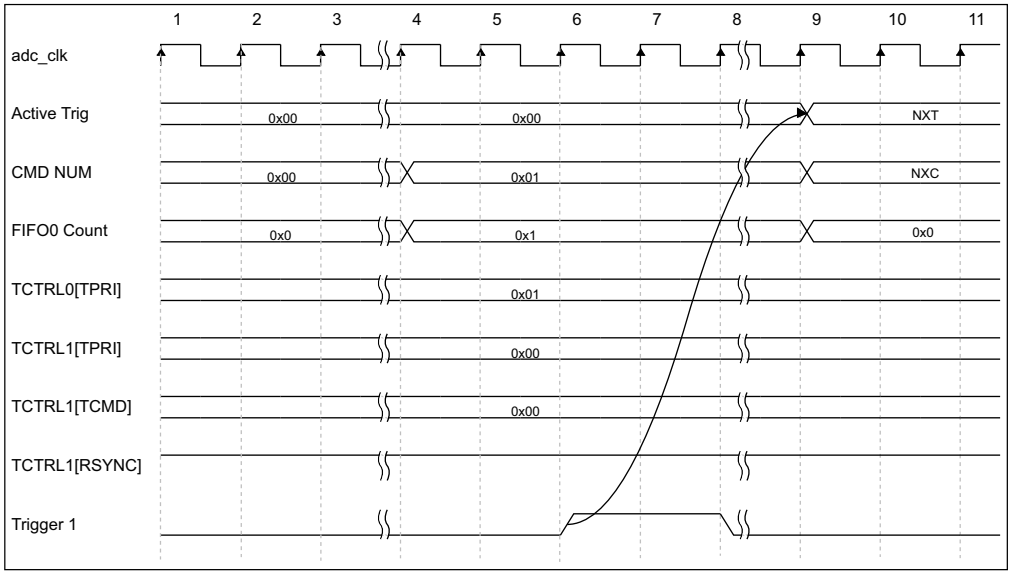


Fig 149. ADC resync example

39.7.5.3 Calibration

The ADC contains a self-calibration function that is required to achieve the specified accuracy.

- Calibration must be run after any reset and before a conversion is initiated.
- If calibration is requested during the middle of a sequence, that sequence will be completed before calibration is initiated.
- If calibration is requested while the ADC is disabled (CTRL[ADCEN] = 0x0), the ADC must first be enabled before the calibration function will run.

Prior to calibration, configure the ADC's clock source and frequency, low power configuration, voltage reference selection, sample time, and high speed configuration according to the application's clock source availability and requirements. Once the

calibration function begins running, it will not be interrupted until all of the CAL_GBR, CAL_GAR, and GCR registers are calculated. The programmer must calculate the GCR register value before calibration can complete.

Remark: Improved accuracy can be achieved during the calibration routine by averaging multiple conversions.

CTRL[CAL_AVGS] will be used during the calibration routine to determine how many samples are averaged together. If the application uses the ADC in a wide variety of configurations, select the configuration for which the highest accuracy is required, or multiple calibrations can be done for the different configurations.

Remark: Prior to running calibration, it is recommended to run CALOFS calibration to calculate the ADC comparator offset voltage.

CALOFS can be calculated by first setting CTRL[CALOFS] to 0b1 and waiting for OFSTRIM to be updated with the offset values. When the CALOFS routine is completed, the STAT[CAL_RDY] bit will be set to 0b1. Wait until STAT[CAL_RDY] is asserted prior to requesting calibration with CAL_REQ.

To initiate calibration, the user sets the CTRL[CAL_REQ] bit and calibration will automatically begin. Once set, the CAL_REQ bit will remain set until the CAL routine has been accepted by the ADC. After this CAL_REQ will automatically clear.

During the CAL routine, registers GCCa[GAIN_CAL] will be written and flagged with GCCa[RDY]. The GCCa[GAIN_CAL] values must be utilized as arguments into the gain offset function.

For the calibration sequence to complete, generate the GCRa[GCALR] gain offset values using the following procedure:

1. Run the CALOFS calibration routine by asserting CTRL[CALOFS] to 0b1 with the number of averages controlled by CTRL[CAL_AVGS].
2. Initiate the ADC to run a calibration routine. Assert CTRL[CAL_REQ] to 0b1 with the number of averages controlled by CTRL[CAL_AVGS].
3. Poll the GCCa[RDY] flags until they are asserted. It will indicate the ADC calibration data has been calculated.
4. Read the GCCa[GAIN_CAL] registers and store these value for use in later calculations.
5. Calculate gain_offset = (131072)/(131072-GCCa[GAIN_CAL]). It will result in a floating point value somewhere within the range of 1 to 2.
6. Round the fractional component of each gain_offset to 16-bits, and write these values to the GCRa[GCALR] registers.
7. Once GCRa[GCALR] contains the 16-bit fractional result from gain_offset, set the GCRa[RDY] flag to indicate this value is valid.

After completing step 7, the Auto-Calibration sequence commences, and the STAT[CAL_RDY] flag is set. This flag remains set until the user resets the system, or requests a new auto-calibration sequence.

When STAT[CAL_RDY] is set, this will enable the ADC to run in calibrated mode. Each conversion will use a combination of linearity and gain calibration results to correct SAR data.

Calibration conversion latency is required to process each sample. However, due to the pipelined nature of data and control sequences, each conversion can still be initiated without experiencing this calibration delay. Figure 150 shows the calibration data calculation and its utilization. The left portion of this image represents the calibration sequencing, and the right portion represents how calibration data is used.

<insert calibration_datapath_v1.1.svg, caption “ADC Calibration Sequence”>

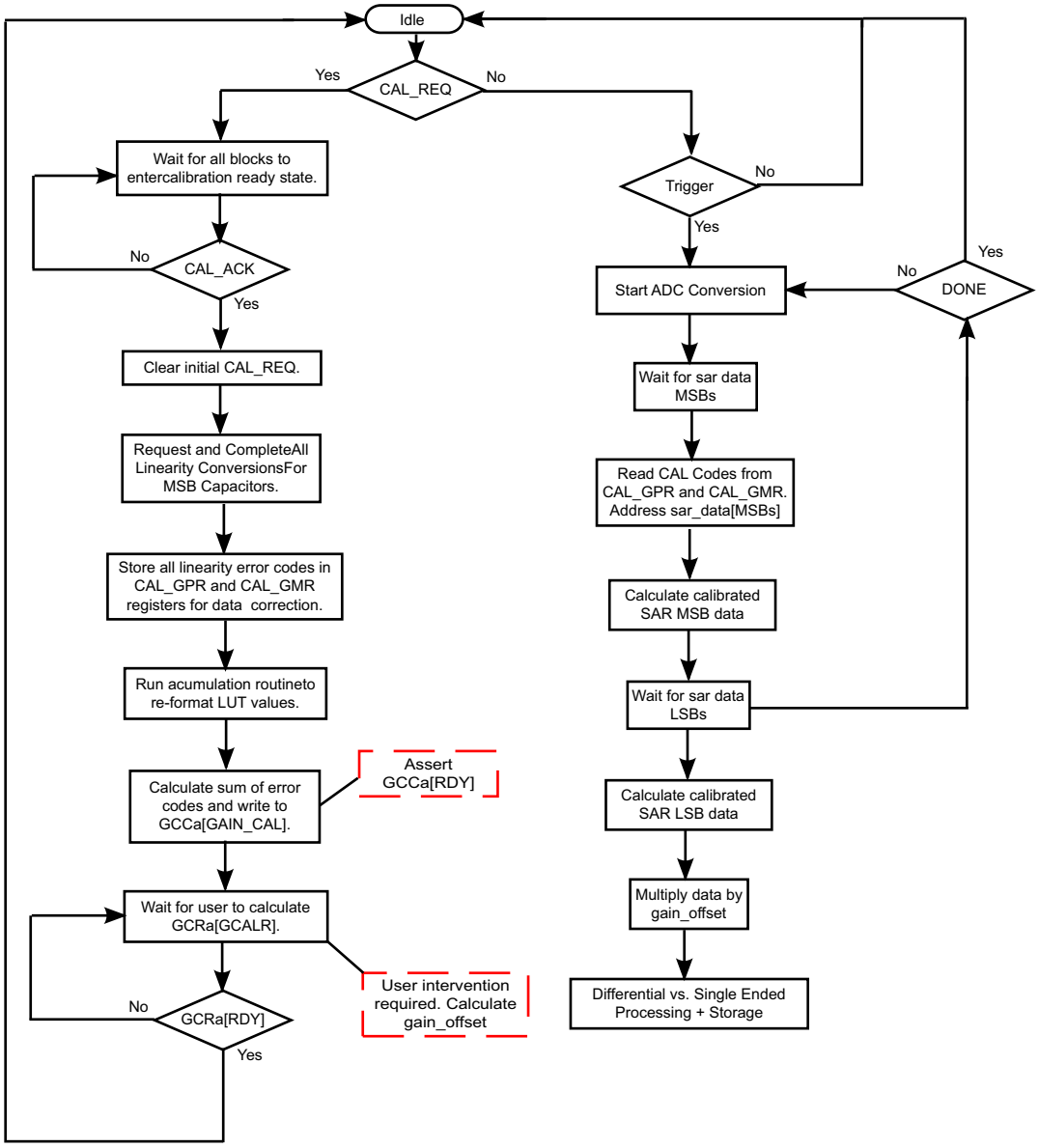


Fig 150. ADC calibration sequence

39.7.6 Temperature sensor

The ADC has a dedicated input channel for an on-chip temperature sensor. It is mapped on channel 26.

To calculate the temperature, the ADC must be configured to run a specific sequence of steps:

1. Channel 26 corresponds to the temperature sensor. To use temperature sensor, the maximum $f_{clk(ADC)}$ (ADC input clock frequency) is 6 MHz.
2. Configure a command register to sample the temperature sensor channel. CMDL[ADCH] = Temperature Sensor Channel.
3. The command must be programmed with the following parameters: CMDL[CTYPE] = 0x2, CMDH[AVGS] = 0x7, CMDH[LOOP] = 0x3, CMDH[LWI] = 0x0, CMDH[STS] = 0x7, CMDH[CMPEN] = 0x0.
4. Configure a trigger control register to associate it with the temperature sensor command: TCTRL[TCMD] = CMD.
5. Trigger a conversion to run the command associated with TCTRL.

After running the temperature sensor command, four results (TS[1:4]) are written to the FIFO selected with TCTRL[FIFO_SEL_A]. A specific combination of the lasts two values (TS[3] and TS[4]) allows on chip temperature to be calculated through software.

To convert the temperature sensor conversion results to the on-chip temperature value:

1. Discard the TS1 and TS2 results, and read the last two conversion results (TS[3] and TS[4])].
2. The first data from step 1 (TS[3]) is called vbe1. The second data (TS[4]) is called vbe8.
3. An equation can be used to convert these results into the final temperature sensor reading: A*[alpha*(Vbe8-Vbe1)/(Vbe8 + alpha*(Vbe8-Vbe1))] - B.
4. Depending on the device revision, following Alpha., A and B values are needed to achieve +/- 4 C temperature accuracy. For device revision 0A: Alpha=9.5, A=770 and B = 289.4. For device revision 1B: Alpha= 8.5, A=804 and B = 280.

Remark: For proper ADC operation, the PDEN_AUXBIAS bit in the PDRUNCFG0 register must be set to “0”(powered) and VREF1VENABLE bit in the AUX_BIAS register must be set to "1" (enabled) prior to using ADC peripheral.

Remark: For best ADC performance, force the offset calibration to -16. Set this prior to performing the gain calibration.

39.7.7 Result FIFO operation

The ADC includes two 16 entry FIFOs in which the result of ADC conversions are stored. In addition, a valid indicator bit, the trigger source, the source command and the loop count are also stored along with the data. FCTRLa[FCOUNT] indicates how many valid data words are stored in each RESFIFO.

A programmable watermark threshold supports configurable notification of data availability.

When FCOUNT is greater than FWMARK, the associated RDY flag is asserted.

When FWMIE is set, a watermark interrupt request is issued.

When FWMDE is set, a DMA request is issued.

Reading RESFIFO provides the oldest unread data word entry in the FIFO and decrements FCOUNT.

When FCOUNT falls equal to or below FWMARK, the RDY flag is cleared.

Each FIFO can be emptied by successive reads of RESFIFOa. When the RESFIFOa[VALID] bit is 1 the associated FIFO entry is valid. Reading RESFIFO when the FIFO is empty (when RESFIFOa[VALID] is clear and FCOUNT=0x0) provides an undefined data word.

All FIFOs are reset by writing 0b1 to the CTRL[RSTFIFO] bit.

If the ADC attempts to store a data word to the FIFO when the FIFO is full the FIFO overflow flag (FCTRLa[FOF]) is set.

When FOFIE is set, a overflow interrupt request is issued.

The FOF flag is cleared by writing 1 to FOF.

On overflow events no new data is stored and the data associated with the store that triggered the overflow is lost, the current ADC command (sequence) is aborted and all pending trigger events are discarded.

No new triggers are detected until the overflow flag is cleared.

Conversion results can be steered to any FIFO in the design. TCTRLa[FIFO_SEL_A] and TCTRLa[FIFO_SEL_B] are utilized to determine which FIFO to write the final result. Therefore, depending on which trigger is executing, the results can be steered to different locations. Depending on the type of conversion selected, the FIFO destination register fields will be interpreted differently. During either differential, dual-single-ended mode, or single-ended mode (CMDLa[CTYPE] != 0x3) only one result will be produced. The destination during these modes will be determined from TCTRLa[FIFO_SEL_A].

In dual-single-ended mode, both TCTRLa[FIFO_SEL_A] and TCTRLa[FIFO_SEL_B] will be used to determine the Channel A and Channel B destinations respectively.

39.7.8 Sampling modes

The ADC supports three different sampling modes:

- Differential
- Single-ended
- Dual single-ended

The sampling mode is determined by the currently executing command using the register field CMDLa[CTYPE].

When executing a command in dual single-ended mode, two independent conversion results will be calculated and stored in selectable FIFO destinations.

Note: Command processing is not individually controlled for each independent channel being sampled.

When operating in dual single-ended mode both channels will be sampled and processed simultaneously. The input channels selected for dual single-ended mode must be selected from a pair of inputs (ex. CH0A/CH0B and CH1A/CH1B). If comparisons are enabled in dual single-ended mode, then only the A side channels (CH0A, CH1A and CH2A) will be used for the comparison.

The A side comparison is used to determine if both the A side and B side results are written to the FIFOs. If the A side comparison passes, then both the A side and B side results are stored. If the A side comparison fails, then neither the A side nor B side results will be written to the FIFOs. Inputs to the ADC are paired according to CMDLa[ADCH] to enable differential and dual single-ended operation.

Single ended mode is configurable to allow either A side or B side channels to be sampled. In single ended mode, the results from each conversion can be written to a selectable FIFO using TCTRLa[FIFO_SEL_A].

In differential mode, the final SAR calculation will be equivalent to V(CHA) - V(CHB). If the result is negative, then the value will be stored in sign-extended 2's complement format. TCTRLa[FIFO_SEL_A] will also determine where differential conversion results are stored.

In dual single-ended mode, however, two independent results will be produced. Individual control is provided by using TCTRLa[FIFO_SEL_A] and TCTRLa[FIFO_SEL_B] to select a FIFO destination for both results during this mode.

Note: Both single ended results may be written to the same destination by programming TCTRLa[FIFO_SEL_A] = TCTRLa[FIFO_SEL_B].

In this case, the CH_A results will always be stored before the CH_B results.

39.7.9 Compare function

After the input is sampled and converted and any averaging iterations are performed, the CMDHa[CMPEN] field guides operation of the automatic compare function to optionally only store when the compare operation is true. There are multiple options on command sequencing related to the compare function. See Table 754 and Table 755.

Note: The latency is added to the end of a compare until true conversion to resolve the next command or loop in a sequence. This latency is necessary to calibrate the SAR data before resolving the result of a comparison. it will always be ≤ 5 ADC clock cycles.

Not all command buffers have an associated compare value register. The compare function is only available on command buffers that have a corresponding compare value register.

Table 754. Compare modes

CMDHa[CMPEN]	Compare function	Description
0b00	Compare disabled	Do not perform compare operation. Always store the conversion result to the FIFO.
0b01	Reserved	
0b10	Store on true	Perform compare operation. Store conversion result to FIFO at end of averaging only if compare is true. If compare is false do not store the result to the FIFO. In either the true or false condition, the LOOP setting is considered and increments the LOOP counter before deciding whether the current command has completed or additional LOOP iterations are required.
0b11	Repeat compare until true	Perform compare operation. Store conversion result to FIFO at end of averaging only if compare is true. Once the true condition is found, the LOOP setting is considered and increments the LOOP counter before deciding whether the current command has completed or additional LOOP iterations are required. If the compare is false, do not store the result to the FIFO. The conversion is repeated without consideration of LOOP setting and does not increment the LOOP counter.

Depending on CVa[CVH] and CVa[CVL] values programmed, the compare operation checks whether the result is less than, greater than, or if the result falls within or outside a range determined by two compare values. The compare values are used as described in Table 755.

Table 755. Compare operations

CVa[CVL] vs. CVa[CVH]	Operation	Description
set CVL< CVH	Outside range (General form)	Compare true if the result is less than CVL value or greater than CVH value.
set CVH to max value/set CVL to compare point	Less than	Compare true if the result is less than CVL value.
set CVL to min value/set CVL to compare point	Greater than	Compare true if the result is greater than CVH value.
set CVL > CVH	Inside range	Compare true if the result is less than CVL value and greater than CVH value

Remark: In low power modes, where the ADC continues to operate, the compare function can monitor the voltage and only wake up the device when the compare condition is met.

39.7.10 Calibration general A-side and B-side widths

The general calibration value registers CAL_GARa and CAL_GBRa have non-uniform widths.

Remark: These values can only be written in a single access, byte accesses aren't supported.

Table 750 represents the bit widths of each register in both CAL_GAR and CAL_GBR:

Table 756. Calibration general widths

Element CAL_GxR[N]	Width (Bits)
N=0x00	11
N=0x01	12
N=0x02	13

Table 756. Calibration general widths *…continued*

Element CAL_GxR[N]	Width (Bits)
N=0x03	13
N=0x04	14
N=0x05	14
N=0x06	14
N=0x07	14
N=0x08	15
N=0x09	15
N=0x0A	15
N=0x0B	15
N=0x0C	15
N=0x0D	15
N=0x0E	15
N=0x0F	15
N=0x10	16
N=0x11	16
N=0x12	16
N=0x13	16
N=0x14	16
N=0x15	16
N=0x16	16
N=0x17	16
N=0x18	16
N=0x19	16
N=0x1A	16
N=0x1B	16
N=0x1C	16
N=0x1D	16
N=0x1E	16
N=0x1F	16
N=0x20	11

39.7.11 Test operation

When TST[TESTEN] is set, the ADC will begin the BIST routine automatically. Any conversions in progress are completed before launching the BIST sequence request. The test operation involves performing 67 tests on both the M and P side capacitor arrays in parallel, with the results from each test measurement being written into FIFO0 (P-Side) and FIFO1 (M-side). Both FIFOs must be read while this test is running to make room for results from later tests. A watermark level can be configured for each FIFO to trigger an interrupt upon filling to a select number of results. This test does not allow the FIFOs to overflow and instead stalls the progression of tests if either of the FIFOs reaches full.

Averaging multiple test conversions to produce a single test result is supported via the CTRL[CAL_AVGS] control field. TST[TESTEN] is cleared by hardware upon completion of the BIST sequence.

The status flag STAT[CAL_RDY] can be monitored to determine when the BIST routine has completed.The CALOFS can be run prior to kicking off the BIST sequence to calibrate for offset. Leaving CALOFS set to the default value of zero has no effect on BIST results.

When TST[FOFFP] is set, an offset is forced on the plus side (A-side) DAC during the compare phase of each conversion cycle.

When TST[FOFFM] is set, an offset is forced on the minus side (B-side) DAC during the compare phase of each conversion cycle.

This feature is used in software based BIST testing of the ADC hard block. The ADC still requires the normal command setup (in single-ended configuration) and triggering steps and executes any configured delays, but the value stored in the RESFIFO is the conversion result of the selected input voltage minus the forced offset voltage. The nominal shifted offset count is 64.

Note: Forcing offset should be done independent of executing any of the targeted test modes. Do not set FOFFP and/or FOFFM when TESTEN is set.

40.1 How to read this chapter

The analog comparator is available on all LPC55S6x/LPC55S2x/LPC552x parts.

40.2 Features

- Selectable external inputs can be used as either the positive or negative input of the comparator.
- Voltage ladder source selectable between the supply, multiplexing between internal vbat_pmu and comp_vi_ref (pad).
- Voltage ladder can be separately powered down when not required (vref_int block is automatically enabled - comp_vref_enable = 1 - as soon as PMUX or NMUX input 0 is selected).
- 32-stage voltage ladder can be used as either the positive or negative input of the comparator.
- Interrupt capability. Can be a wake up source in deep-sleep and power-down low power modes

40.3 Basic configuration

Configure the analog comparator using the following registers:

- In the AHBCLKCTRL2 register, set bit 2, see Table 57 to enable the clock to the register interface.
- You can enable or disable the power to the analog comparator through the PDRUNCFG register, see Table 311.
- Clear the analog comparator peripheral reset using the AHBCLKCTRLSET2 register, see Table 60.
- The analog comparator interrupt is connected to interrupt #24 in the NVIC.
- Configure the analog comparator pin functions through IOCON. See Chapter 16 “LPC55S6x/LPC55S2x/LPC552x General Purpose I/O (GPIO)”.

40.4 Pin description

The analog comparator reference voltage, the inputs, and the output are assigned to external pins through IOCON. The comparator inputs and the reference voltage are fixed-pin functions that must be enabled through IOCON and can only be assigned to special pins on the package.

See Chapter 16 “LPC55S6x/LPC55S2x/LPC552x General Purpose I/O (GPIO)” to assign the analog comparator output to any pin on the LPC81x package.

See Table 760 to enable the analog comparator inputs and the reference voltage input.

Table 757. Analog comparator pin description

Function	Type	Pin	Description	SWM register
ACMP0_A	I	PIO0_0	Comparator input 1	PMC.COMP
ACMP0_B	I	PIO0_9	Comparator input 2	PMC.COMP
ACMP0_C	I	PIO0_18	Comparator input 3	PMC.COMP
ACMP0_D	I	PIO1_14	Comparator input 4	PMC.COMP
CMP0_OUT	O	PIO0_1, PIO0_29	Comparator output	-
ACMPvref	I	PIO1_19	External reference voltage source for 32-stage Voltage Ladder.	-

40.5 General description

The analog comparator can compare voltage levels on external pins and internal voltages.

The comparator has 5 inputs multiplexed separately to its positive and negative inputs. The multiplexers are controlled by the comparator register. See Table 760

Any input can be selected on the P side of the comparator (by COM[PMUX]) and compared to any input on the N side of the comparator (by COMP[NMUX]).

40.5.1 Comparator modes

The analog comparator supports both Standard and Low Power mode. Comparator mode can be selected by LOWPOWER value.

In Standard mode (LOWPOWER = 0), comparator delay is typically 15 μs in low overdrive configuration (inputs voltage difference of 10 mV). Overdrive mode refers to the comparator input voltage difference.

In Low Power mode (LOWPOWER = 1), the comparator current consumption can be reduced to 360 nA (typical, in case voltage ladder source is not selected) at the expense of higher comparator delay (95 μs typical in low overdrive configuration). This last mode is suitable for IC low power modes.

Typical comparator delay is 10 us in large overdrive mode with any LOWPOWER setting.

40.5.2 Reference voltages

The voltage ladder can use two reference voltages (VBAT_PMU or $ACMPV_{REF}$). The voltage ladder selects one of 32 steps between the pin voltage and V_{SS} inclusive.

40.5.3 Settling times

After the voltage ladder is powered on, it requires stabilization time until comparisons using it are accurate. Much shorter settling times apply after the VREFINPUT value is changed and when either or both voltage sources are changed. Software can deal with these factors by repeatedly reading the comparator output until a number of readings yield the same result.

40.5.4 Interrupts

Interrupt management is set with COMP_INT_CTRL and COMP_INT_STATUS registers Table 761 and Table 762.

The interrupt output comes from edge detection circuitry in this module. Rising edges, falling edges, or both edges can be set in the INT_CTRL field. Interrupt requests are cleared when software writes a 1 to INT_CLEAR. The source can also be selected with INT_SOURCE to use a filtered or unfiltered comparator output.

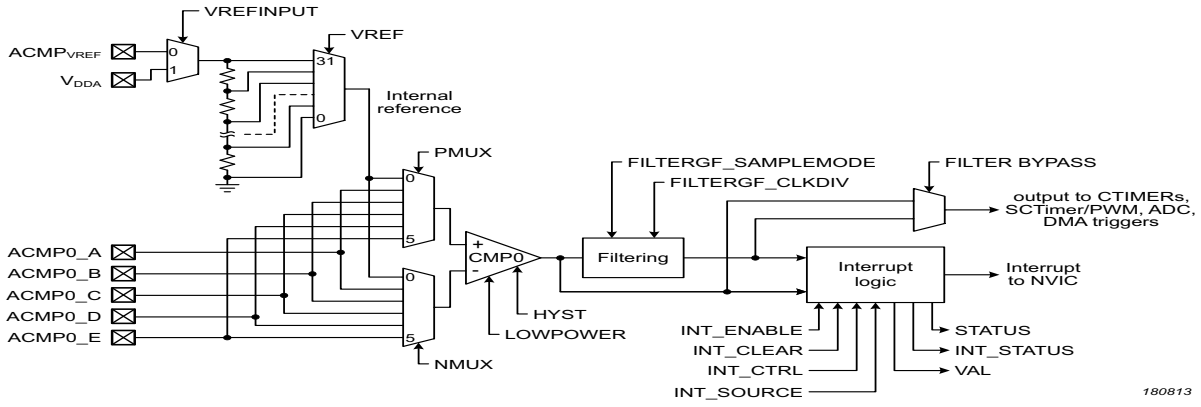


Fig 151. Comparator block diagram

40.5.5 Comparator outputs

The comparator output can be routed to an external pin.The comparator can be used with the bus clock disabled, see Table 57 to save power if the control registers are not required to be written.

The status of the comparator output can be observed through the comparator status register bit (COMP_INT_STATUS). Comparator outputs are connected to the I/O pad and can also be used as trigger inputs to various on-chip peripherals (for example, CTimers, SCTimer/PWM, ADC, DMA controllers).

40.6 Register description

Table 758. Register overview: PMC comparator (base address 0x5002 0000)

Name	Access	Address offset	Description	Reset value	Reference
COMP	R/W	0x050	Comparator control register	0	40.6.1

Table 759. Register overview: SYSCON comparator (base address 0x5000 0000)

Name	Access	Address offset	Description	Reset value	Reference
COMP_INT_CTRL	R/W	0xB10	Comparator interrupt control	0	40.6.2
COMP_INT_STATUS	WO	0xB14	Comparator interrupt status	0	40.6.3

40.6.1 Analog comparator control register

The analog comparator control register enables the comparator, configures the interrupts, and controls the input multiplexers on both sides of the comparator. All bits not shown in Table 760 are reserved and should be written as 0.

Table 760. Analog comparator control register (COMP, offset = 0x50)

Bit	Symbol	Access	Value	Description	Reset value
0		WO		Reserved. Read value is undefined, only zero should be written.	undefined
1	HYST	RW		Hysteris when hyst = ’1’.	0x1
			0	Hysteresis is disable.	
			1	Hysteresis is enable.	
2	VREFINPUT	RW		Dedicated control bit to select input voltage of programmable resistive ladder: Either ACMPvref (PIO1_19) or VDDA (VBAT_PMU).	0x0
			0	Select internal ACMPvref.	
			1	Select VDDA.	
3	LOWPOWER	RW		Low power mode.	0x1
			0	High speed mode.	
			1	Low power mode (Low speed).	
6:4	PMUX	RW		Control word for P multiplexer.	0x0
			0	VREF (See field VREFINPUT).	
			1	PIO0_0.	
			2	PIO0_9.	
			3	PIO0_18.	
			4	PIO1_14.	
9:7	NMUX	RW		Control word for N multiplexer:.	0x0
			0	VREF (See field VREFINPUT).	
			1	Pin P0_0.	
			2	Pin P0_9.	
			3	Pin P0_18.	
			4	Pin P1_14.	
14:10	VREF	RW		Control reference voltage step, per steps of (VREFINPUT/31).	0x0
15		WO		Reserved. Read value is undefined, only zero should be written.	undefined
17:16	FILTERCGF_SAMPLEMODE	RW		Control the filtering of the Analog Comparator output.	0x0
			0	Bypass mode. Filtering is disabled. The raw Analog Comparator output will be passed-through.	
			1	Filter 1 clock period. Any pulse duration shorter than one cycle of the designated filter clock (see FILTERCGF_CLKDIV) will be filtered-out. Pulse widths up to two cycles long may be filtered.	
			2	Filter 2 clock period. Any pulse duration shorter than two cycles of the designated filter clock will be filtered-out. Pulse widths up to three cycles long may be filtered.	
			3	Filter 3 clock period. Any pulse duration shorter than three cycles of the designated filter clock will be filtered-out. Pulse widths up to four cycles long may be filtered.	

Table 760. Analog comparator control register (COMP, offset = 0x50) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
20:18	FILTERCGF_CLKDIV	RW		Filter clock divider. Filter clock equals the Analog Comparator clock divided by 2^FILTERCGF_CLKDIV.	0x0
			0	Filter clock period duration equals 1 Analog Comparator clock period.	
			1	Filter clock period duration equals 2 Analog Comparator clock period.	
			2	Filter clock period duration equals 4 Analog Comparator clock period.	
			3	Filter clock period duration equals 8 Analog Comparator clock period.	
			4	Filter clock period duration equals 16 Analog Comparator clock period.	
			5	Filter clock period duration equals 32 Analog Comparator clock period.	
			6	Filter clock period duration equals 64 Analog Comparator clock period	
			7	Filter clock period duration equals 128 Analog Comparator clock period	
31:21		WO		Reserved. Read value is undefined, only zero should be written.	undefined

40.6.2 Comparator interrupt control register

All interrupts can be managed with the comparator interrupt control register. Rising edges, falling edges, or both edges analog comparator interrupts can be requested. The interrupt request are cleared when software writes a 1 to INT_CLEAR bit.

Table 761. Comparator Interrupt control (COMP_INT_CTRL, offset = 0xB10)

Bit	Symbol	Access	Value	Description	Reset value
0	INT_ENABLE	RW		Analog Comparator interrupt enable control:.	0x0
			1	interrupt enable.	
			0	interrupt disable.	
1	INT_CLEAR	RW		Analog Comparator interrupt clear.	0x0
			0	No effect.	
			1	Clear the interrupt. Self-cleared bit.	
4:2	INT_CTRL	RW		Comparator interrupt type selector.	0x0
			0	The analog comparator interrupt edge sensitive is disabled.	
			2	Analog comparator interrupt is rising edge sensitive.	
			4	Analog comparator interrupt is falling edge sensitive.	
			6	Analog comparator interrupt is rising and falling edge sensitive.	
			1	The analog comparator interrupt level sensitive is disabled.	
			3	Analog Comparator interrupt is high level sensitive.	
			5	Analog Comparator interrupt is low level sensitive.	
			7	The analog comparator interrupt level sensitive is disabled.	

Table 761. Comparator Interrupt control (COMP_INT_CTRL, offset = 0xB10)

Bit	Symbol	Access	Value	Description	Reset value
5	INT_SOURCE	RW		Select which Analog comparator output (filtered our un-filtered) is used for interrupt detection.	0x0
			0	Select Analog Comparator filtered output as input for interrupt detection.	
			1	Select Analog Comparator raw output (unfiltered) as input for interrupt detection. Must be used when Analog comparator is used as wake up source in Power down mode.	
31:6		WO		Reserved. Read value is undefined, only zero should be written.	undefined

40.6.3 Comparator interrupt status register

The comparator interrupt status register provides the status of the interrupt and comparator output.

Table 762. Comparator interrupt status (COMP_INT_STATUS, offset = 0xB14)

Bit	Symbol	Access	Value	Description	Reset value
0	STATUS	RO		Interrupt status BEFORE interrupt enable.	0x0
			0	No interrupt pending.	
			1	Interrupt pending.	
1	INT_STATUS	RO		Interrupt status AFTER interrupt enable.	0x0
			0	No interrupt pending.	
			1	Interrupt pending.	
2	VAL	RO		Comparator analog output.	0x0
			1	P+ is greater than P-.	
			0	P+ is smaller than P-.	
31:3		WO		Reserved. Read value is undefined, only zero should be written.	undefined

41.1 How to read this chapter

The USB full-speed controller is available on all LPC55S6x/LPC55S2x/LPC552x devices.This chapter describes the device functionality of the controller.

41.2 Features

- USB2.0 full-speed device controller supporting crystal-less operation in device mode using the software library example in technical note TN00063.
- Supports ten physical (five logical) endpoints including the control endpoints.
- Supports single and double buffering.
- 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 internally.
- Supports Link Power Management (LPM).

41.3 Basic configuration

Initial configuration of the USB0 device controller:

- Pins: Configure the USB0 pins in the IOCON register block. See Table 333 and Section 41.6 “Pin description”.
- In the AHBCLKCTRL1 register, enable the clock to the USB0D device controller register interface, see Section 4.5.18 “AHB clock control 1”.
- Power: Enable the power to the USB0 PHY by clearing the bit PDEN_USBFSPHY in the PDRUNCFG0 register, see Section 13.4.17 “Power configuration register 0”.
- Port mode configuration: Enable port mode configuration by setting the USB0_HOSTS in the AHBCLKCTRL2 register. See Table 57. Set DEV_ENABLE in Section 42.7.23 “PortMode register” in Port Mode register (offset 0x5C) to enable the device controller on the USB0 port. Once configured, to save power, clear USB0_HOSTS in the AHBCLKCTRL2 register, See Section 4.5.19 “AHB clock control 2”.
- Reset: The USB0 device can be reset by toggling USB0_DEV_RST in PRESETCTRL2.See Section 4.5.9 “Peripheral reset control 2”.
- Interrupts: The USB0 device controller has two interrupt sources allocated in the NVIC interrupt source table: a general interrupt, USB0, and an activity interrupt, USB0_NEEDCLK. See Section 3.4.7 “Interrupt clear pending register 0”. Clear pending interrupts before enabling them.
- Configure the USB0 main clock, see Section 41.4.7 “Clocking”).
- Configure the USB0 wake-up signal, see Section 41.8.6 “USB0 wake-up”) if needed.

41.4 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 eight logical or 16 physical endpoints including control endpoint. 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 USB0 device controller enables full-speed (12 Mb/s) data exchange with a USB host controller.

Figure 152 shows the block diagram of the USB0 device controller.

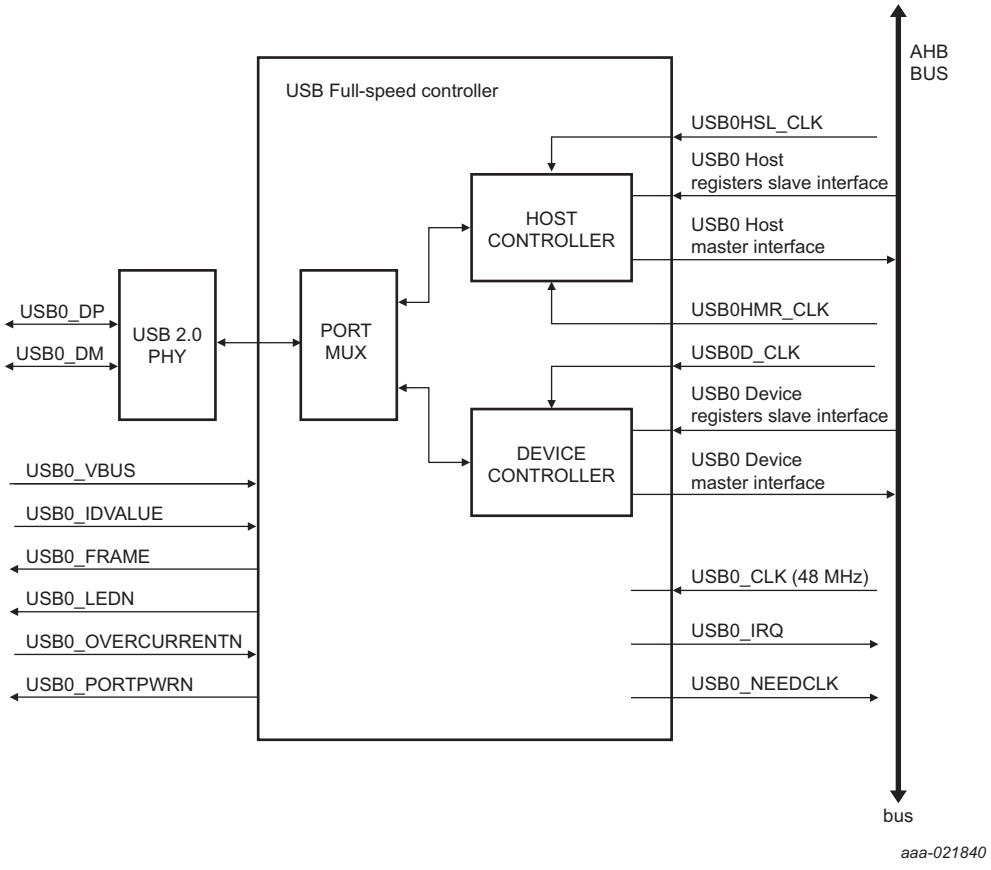


Fig 152. USB full-speed host/device controller block diagram

41.4.1 USB0 software interface

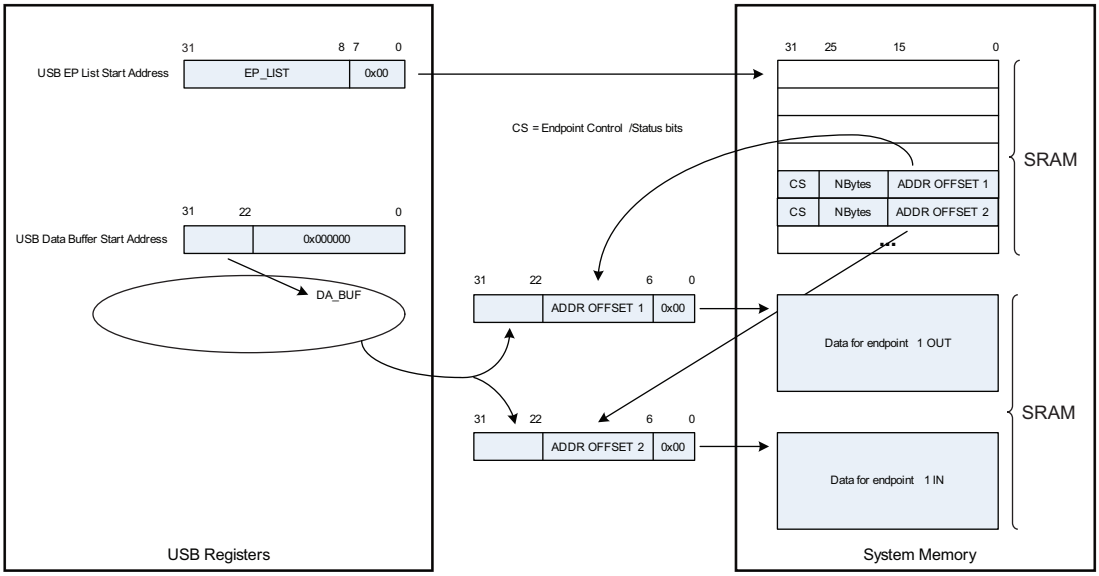


Fig 153. USB0 software interface

41.4.2 Fixed endpoint configuration

Table 763 shows the supported endpoint configurations. The packet size is configurable up to the maximum value for each type of end point.

Table 763. 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

41.4.3 Soft connect

The softConnect signal is implemented internally. An external pull-up resistor between USB_DP and VDD is not necessary. Software can control the pull-up 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 VDD through an internal 1.5 KOhm pull-up resistor.

41.4.4 Interrupts

The USB controller has two interrupt lines, a general USB interrupt (USB0) and a USB activity wake-up interrupt (USB0_NEEDCLK). See Table 20. A general 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). See Section 41.7.9 “USB0 interrupt status register” and Section 41.7.10 “USB0 interrupt enable register”.

41.4.5 Suspend and resume

The USB protocol insists on power management by the USB device. It 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 (device-initiated wake up). To initiate remote wake-up, the software on the device must enable all clocks and clear the DSUS in DEVCMDSTAT bit. It will cause the hardware to generate a remote wake-up signal upstream.

The USB controller supports Link Power Management (LPM).

The assertion of the 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 a suspended state.

If there is no activity for the next 2 ms, the USB0 DEV_NEEDCLK signal will go LOW. This indicates that the USB main clock can be switched off.

When any activity is detected on the USB bus, the DEV_NEEDCLK signal is activated. This process is fully combinatorial and therefore, USB main clock is not required to activate the DEV_NEEDCLK signal. See Section 41.8.6 “USB0 wake-up” for more details on waking up from suspend mode.

41.4.6 Frame toggle output

The USB0_FRAME output pin reflects the 1 kHz clock derived from the incoming start of frame tokens sent by the USB host.

41.4.7 Clocking

The USB0 device controller has the following clock connections:

- USB main clock: The USB main clock is a 48 MHz clock used for USB functions (see Section 4.5.40 “USB0 clock source select register” and Section 4.5.55 “USB0 full-speed clock divider register”. If the FRO is used as the USB clock source, it can be configured to adjust automatically to the USB bus rate, see Section 11.5.4 “FRO192M control register”.
- CPU clock: The minimum frequency of the CPU clock is 12 MHz when the USB device controller is receiving or transmitting USB packets.
- PLL (PLL1) is added to provide 48 MHz accurate clock-source for USB-FS host.

41.5 Separate USB PHY power

A separate USB PHY power pad is to be added. There will be a bonding option to tie this to power without using a device pin. As illustrated below, when USB power is pinned out, customer will be able to:

- Tie USB PHY power directly to 3.3V on the board.
- Tie USB PHY power to a pull down resistor (if needed), to be brought up by USB power when that is plugged in.

Include from VISIO file “USB_FS”

41.6 Pin description

The device controller can access one USB0 port.

Table 764. USB0 device pin description

Name	Port pin	IOCON function/Mode	Direction	Description
USB0_VBUS	PIO0_22 PIO1_11	PIO0_22, function 7 Mode: inactive PIO1_11, function 4 Mode: inactive	I	USB VBUS status input. When this function is not enabled via its corresponding IOCON register, it is driven LOW internally.
USB0_DP	-	-	I/O	Positive differential data.
USB0_DM	-	-	I/O	Negative differential data.
USB0_IDVALUE	PIO0_26	PIO0_26, function 7 Mode: pull-up	I	A-device (host role) or B-device (peripheral role) indication. Enable this function when using a micro USB receptacle to identify whether a micro-A or micro-B plug is inserted. When enabled, the pull-up on the corresponding port pin should be enabled.
USB0_FRAME	PIO1_13	PIO1_13, function 5 Mode: inactive	O	1 kHz clock derived from the incoming Start of Frame tokens sent by the USB host. It is an optional function.
USB0_LEDN	PIO1_14	PIO1_14, function 5 Mode: inactive	O	USB connection indication. It is an optional function.
USB0_PORTPWRN	-	-	-	Host only function.
USB0_OVERCURRENTN	-	-	-	Host only function.

41.7 Register description

Table 765. Register overview: USB0 (base address: 0x4008 4000)

Name	Access	Offset	Description	Reset value	Section
DEVCMDSTAT	R/W	0x000	USB device command/status register.	0x00000800	41.7.1
INFO	RO	0x004	USB info registe.r	0x01060000	41.7.2
EPLISTSTART	R/W	0x008	USB EP command/status list start address.	0	41.7.3
DATABUFSTART	R/W	0x00C	USB data buffer start address.	0	41.7.4
LPM	R/W	0x010	USB link power management register.	0	41.7.5
EPSKIP	R/W	0x014	USB endpoint skip.	0	41.7.6
EPINUSE	R/W	0x018	USB endpoint buffer in use.	0	41.7.7
EPBUFCFG	R/W	0x01C	USB endpoint buffer configuration register.	0	41.7.8
INTSTAT	R/W	0x020	USB interrupt status register.	0	41.7.9
INTEN	R/W	0x024	USB interrupt enable register.	0	41.7.10
INTSETSTAT	R/W	0x028	USB set interrupt status register.	0	41.7.11
EPTOGGLE	R/W	0x034	USB endpoint toggle register.	0	41.7.12

41.7.1 USB0 device command/status register

This register contains all the fields to control the behavior of the full-speed USB device.

Table 766. USB0 device command/status register (DEVCMDSTAT, offset 0x000)

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, the software must program the new address before completing the status phase of the SetAddress control request.	0	R/W
7	DEV_EN	-	USB device enable. If this bit is set, the hardware will start responding on packets for function address DEV_ADDR.	0	R/W
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 hardware. The software must clear this bit by writing a 1. If this bit is 0, the hardware will handle the tokens to the CTRL EP0 as indicated by the CTRL EP0 IN and OUT data information programmed by software.	0	R/W1C
9	FORCE_ NEEDCLK		Forces the NEEDCLK output to always be ON.	0	R/W
		0	USB_NEEDCLK has normal function.		
		1	USB_NEEDCLK always 1. Clock will not be stopped in case of suspend.		
10	-	-	Reserved.	-	-
11	LPM_SUP		LPM supported:	1	R/W
		0	LPM not supported.		
		1	LPM supported.		

Table 766. USB0 device command/status register (DEVCMDSTAT, offset 0x000) *...continued*

Bit	Symbol	Value	Description	Reset value	Access
12	INTONNAK_AO		Interrupt on NAK for interrupt and bulk OUT EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
13	INTONNAK_AI		Interrupt on NAK for interrupt and bulk IN EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
14	INTONNAK_CO		Interrupt on NAK for control OUT EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
15	INTONNAK_CI		Interrupt on NAK for control IN EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
16	DCON	-	Device status - 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.	0	R/W
17	DSUS	-	Device status - suspend. The suspend bit indicates the current suspend state. It is set to 1 when the device has not seen any activity on its upstream port for more than 3 ms. 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, writing a 0 has no effect. Writing a 1 never has an effect.	0	R/W
18	-	-	Reserved.	-	-
19	LPM_SUS	-	Device status - LPM suspend. This bit represents the current LPM suspend state. It is set to 1 by hardware 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 0 to this bit, the device will generate a remote walk-up. Software can only write a 0 to this bit when the LPM_REWP bit is set to 1. Hardware resets this bit when it receives a host initiated resume. Hardware only updates the LPM_SUS bit when the LPM_SUPP bit is equal to 1.	0	R/W
20	LPM_REWP	-	LPM remote wake-up enabled by USB host. Hardware sets this bit to one when the bRemoteWake bit in the LPM extended token is set to 1. Hardware 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
23:21	-	-	Reserved.	-	-
24	DCON_C	-	Device status - connect change. The connect change bit is set when the pull-up resistor of the device is disconnected because VBUS disappeared. The bit is reset by writing a 1 to it.	0	R/W1C

Table 766. USB0 device command/status register (DEVCMDSTAT, offset 0x000) *...continued*

Bit	Symbol	Value	Description	Reset value	Access
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	R/W1C
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	R/W1C
27	-	-	Reserved.	-	-
28	VBUS_ DEBOUNCED	-	This bit indicates if VBUS is detected or not. The bit raises immediately when VBUS becomes high. It drops to 0 if VBUS is low for at least 3 ms. If this bit is high and the DCon bit is set, the hardware will enable the pull-up resistor to signal a connect.	0	RO
31:29	-	-	Reserved.	-	-

41.7.2 USB0 info register

Table 767. USB0 info register (INFO, offset 0x004)

Bit	Symbol	Value	Description	Reset value	Access
10:0	FRAME_NR	-	Frame number. It 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	R/W
		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.		

Table 767. USB0 info register (INFO, offset 0x004) *…continued*

Bit	Symbol	Value	Description	Reset value	Access
15	-	-	Reserved.	0	RO
23:16	MINREV	-	Minor revision.	0x01	RO
31:24	MAJREV	-	Major revision.	0x05	RO

41.7.3 USB0 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 hard coded to 0 because the list must start on a 256 byte boundary. Bits 31 to 8 can be programmed by software.

Table 768. USB0 EP command/status list start address (EPLISTSTART, offset 0x008)

Bit	Symbol	Description	Reset value	Access
7:0	-	Reserved	0	RO
19:8	EP_LIST_PRG	Programmable portion of the USB EP Command/Status List address.		
31:8	EP_LIST_FIXED	Fixed portion of USB EP Command/Status List address.	0	R/W

41.7.4 USB0 data buffer start address

This register indicates the page of the AHB address where the endpoint data can be located. The 22 LSBs are fixed to 0 so that the location resides on a 4 MB boundary. The start address of each individual endpoint's buffer is an offset to the data buffer start address. The buffer address of the endpoint is set using the address offset field of the endpoint's corresponding entry in the endpoint command/status list. See section Section 41.8.1 “Endpoint command/status list”.

Table 769. USB0 data buffer start address (DATABUFSTART, offset 0x00C)

Bit	Symbol	Description	Reset value	Access
21:0	-	The fixed portion of the data buffer start address.	0	RO
31:22	DA_BUF	Programmable portion of the data buffer start address.	0	R/W

41.7.5 USB0 link power management register

Table 770. Link power management register (LPM, offset 0x010)

Bit	Symbol	Description	Reset value	Access
3:0	HIRD_HW	Host initiated resume duration - HW. It is the HIRD value from the last received LPM token.	0	RO

Table 770. Link power management register (LPM, offset 0x010) *...continued*

Bit	Symbol	Description	Reset value	Access
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 1 and LPM supported bit is set to 1, the hardware will return a NYET handshake on every LPM token it receives. If LPM supported bit is set to 1 and this bit is 0, the hardware will return an ACK handshake on every LPM token it receives. If software has data still pending and LPM is supported, it must set this bit to 1.	0	R/W
31:9	-	Reserved.	-	-

41.7.6 USB0 endpoint skip

Table 771. USB0 endpoint skip (EPSKIP, offset 0x014)

Bit	Symbol	Description	Reset value	Access
11:0	SKIP	Endpoint skip: Writing 1 to one of these bits will indicate to hardware that it must deactivate the buffer assigned to this endpoint and return control back to the software. When hardware 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, hardware will only clear the active bit of the buffer indicated by the EPINUSE bit.	0	R/W
31:12	-	Reserved.	-	-

41.7.7 USB0 endpoint buffer in use

Table 772. USB0 endpoint buffer in use (EPINUSE, offset 0x018)

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved. Fixed to 0 because the control endpoint 0 is fixed to single buffering for each physical endpoint.	0	RO
11:2	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
31:12	-	Reserved.	-	-

41.7.8 USB0 endpoint buffer configuration

Table 773. USB0 endpoint buffer configuration (EPBUFCFG, offset 0x01C)

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved. Fixed to 0 because the control endpoint 0 is fixed to single buffering for each physical endpoint.	0	RO
11:2	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), hardware will toggle the EPINUSE bit when it clears the active bit for the buffer.	0	R/W
31:12	-	Reserved.	-	-

41.7.9 USB0 interrupt status register

Table 774. USB0 interrupt status register (INTSTAT, offset 0x020)

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 0 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 1 to it.	0	R/W1C
1	EP0IN	Interrupt status register bit for the control EP0 IN direction. This bit will be set if NBytes transitions to 0 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 1 to it.	0	R/W1C
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 0 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 1 to it.	0	R/W1C
3	EP1IN	Interrupt status register bit for the EP1 IN direction. This bit will be set if the corresponding active bit is cleared by hardware. This is done in case the programmed NBytes transitions to 0 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 1 to it.	0	R/W1C
4	EP2OUT	Interrupt status register bit for the EP2 OUT direction. This bit will be set if the corresponding active bit is cleared by hardware. This is done in case the programmed NBytes transitions to 0 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 1 to it.	0	R/W1C

Table 774. USB0 interrupt status register (INTSTAT, offset 0x020) *…continued*

Bit	Symbol	Description	Reset value	Access
5	EP2IN	Interrupt status register bit for the EP2 IN direction. This bit will be set if the corresponding active bit is cleared by hardware. This is done in case the programmed NBytes transitions to 0 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 1 to it.	0	R/W1C
6	EP3OUT	Interrupt status register bit for the EP3 OUT direction. This bit will be set if the corresponding active bit is cleared by hardware. This is done in case the programmed NBytes transitions to 0 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 1 to it.	0	R/W1C
7	EP3IN	Interrupt status register bit for the EP3 IN direction. This bit will be set if the corresponding active bit is cleared by hardware. This is done in case the programmed NBytes transitions to 0 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 1 to it.	0	R/W1C
8	EP4OUT	Interrupt status register bit for the EP4 OUT direction. This bit will be set if the corresponding active bit is cleared by hardware. This is done in case the programmed NBytes transitions to 0 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 1 to it.	0	R/W1C
9	EP4IN	Interrupt status register bit for the EP4 IN direction. This bit will be set if the corresponding active bit is cleared by hardware. This is done in case the programmed NBytes transitions to 0 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 1 to it.	0	R/W1C
29:10	-	Reserved.	-	-
30	FRAME_INT	Frame interrupt. This bit is set to 1 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 1 to it.	0	R/W1C
31	DEV_INT	Device status interrupt. This bit is set by hardware when one of the bits in the device status change register are set. Software can clear this bit by writing a 1 to it.	0	R/W1C

41.7.10 USB0 interrupt enable register

Table 775. USB0 interrupt enable register (INTEN, offset 0x024)

Bit	Symbol	Description	Reset value	Access
11:0	EP_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a hardware interrupt is generated on the interrupt line.	0	R/W
29:12	-	Reserved.	-	-
30	FRAME_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a hardware interrupt is generated on the interrupt line.	0	R/W
31	DEV_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a hardware interrupt is generated on the interrupt line.	0	R/W

41.7.11 USB0 set interrupt status register

Table 776. USB0 set interrupt status register (INTSETSTAT, offset 0x028)

Bit	Symbol	Description	Reset value	Access
11:0	EP_SET_INT	If software writes a 1 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
29:12	-	Reserved.	-	-
30	FRAME_SET_INT	If software writes a 1 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 1 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

41.7.12 USB0 endpoint toggle

Table 777. USB0 endpoint toggle (EPTOGGLE, offset 0x034)

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.	-	-

41.8 Functional description

41.8.1 Endpoint command/status list

Figure 154 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 Figure 154.

USB EP Command/Status FIFO start

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	EP14 OUT Buffer 0 NBytes										EP14 OUT Buffer 0 Address Offset																	0xE0
A	D	S	TR	RF TV	T	EP14 OUT Buffer 1 NBytes										EP14 OUT Buffer 1 Address Offset																	0xE4
A	D	S	TR	RF TV	T	EP14 IN Buffer 0 NBytes										EP14 IN Buffer 0 Address Offset																	0xE8
A	D	S	TR	RF TV	T	EP14 IN Buffer 1 NBytes										EP14 IN Buffer 1 Address Offset																	0xEC

Fig 154. Endpoint command/status list (see Table 778)

Table 778. Endpoint command/status bit definitions

Symbol	Access	Description
A	R/W	Active The buffer is enabled. Hardware 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 1 to the corresponding “skip” bit in the USB endpoint skip register. Hardware can only write this bit to 0. It will do this when it receives a short packet or when the NBytes field transitions to 0 or when software has written a 1 to the “skip” bit.
D	R/W	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 0.
S	R/W	Stall 0: The selected endpoint is not stalled. 1: The selected endpoint is stalled. The active bit has always a higher priority than the stall bit. This means that a Stall handshake is only sent when the active bit is 0 and the stall bit is 1. Software can only modify this bit when the active bit is 0.

Table 778. Endpoint command/status bit definitions *…continued*

Symbol	Access	Description
TR	R/W	Toggle reset When software sets this bit to 1, the hardware will set the toggle value equal to the value indicated in the “toggle value” (TV) bit. For the control endpoint 0, 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 0 when the endpoint is reset.
RF / TV	R/W	Rate feedback mode / Toggle value For bulk endpoints and isochronous endpoints this bit is reserved and must be set to 0. For the control endpoint 0 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 0 for all data packets. When the interrupt endpoint is in ‘rate feedback mode’, the TR bit must always be set to 0.
T	R/W	Endpoint type 0: Generic endpoint. The endpoint is configured as a bulk or interrupt endpoint. 1: Isochronous endpoint.
NBytes	R/W	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	R/W	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 0, the hardware 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 eight bytes, it will only write the first eight bytes to memory. A USB compliant host must never send more than eight 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 155.

41.8.2 Control endpoint 0

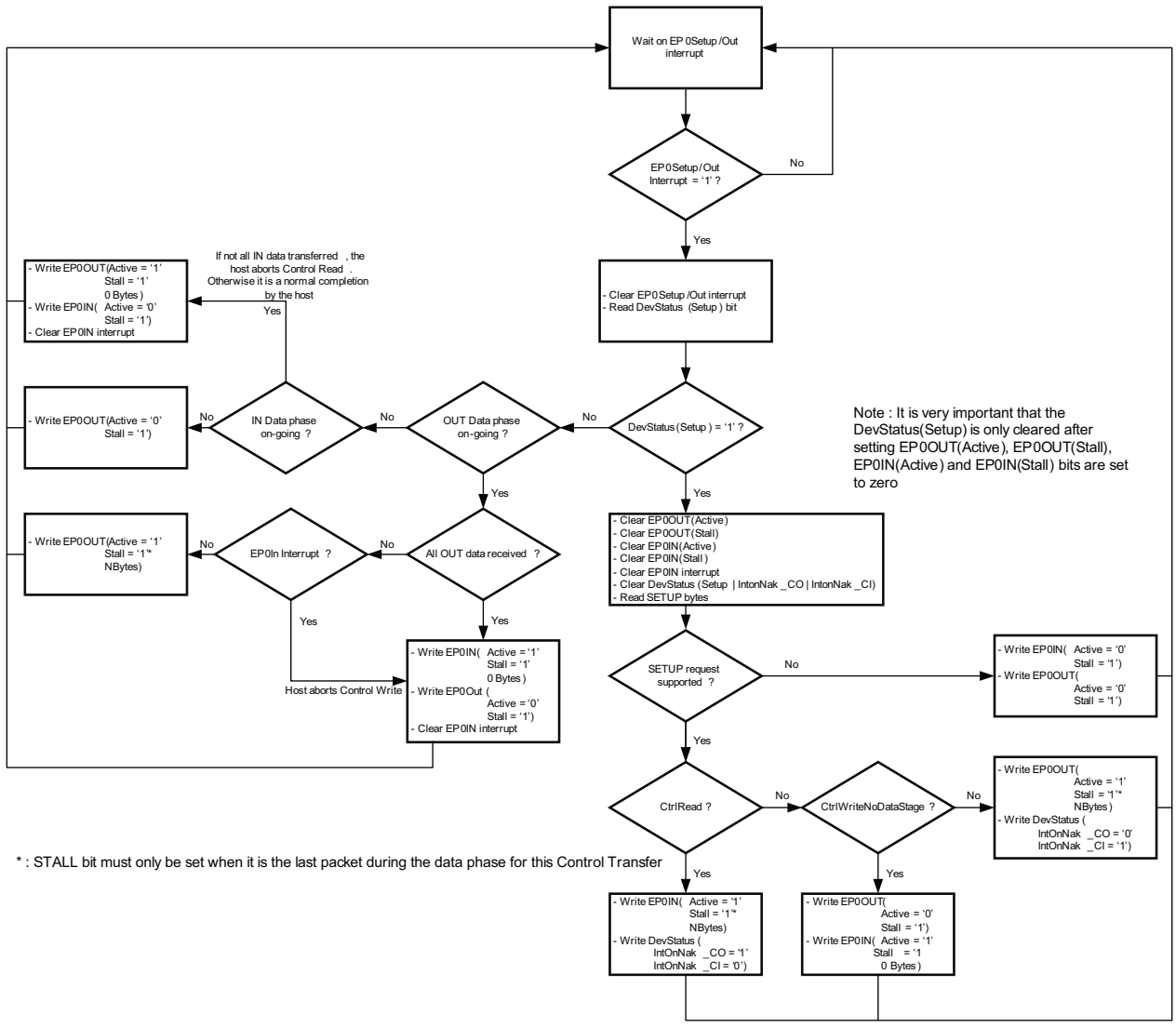
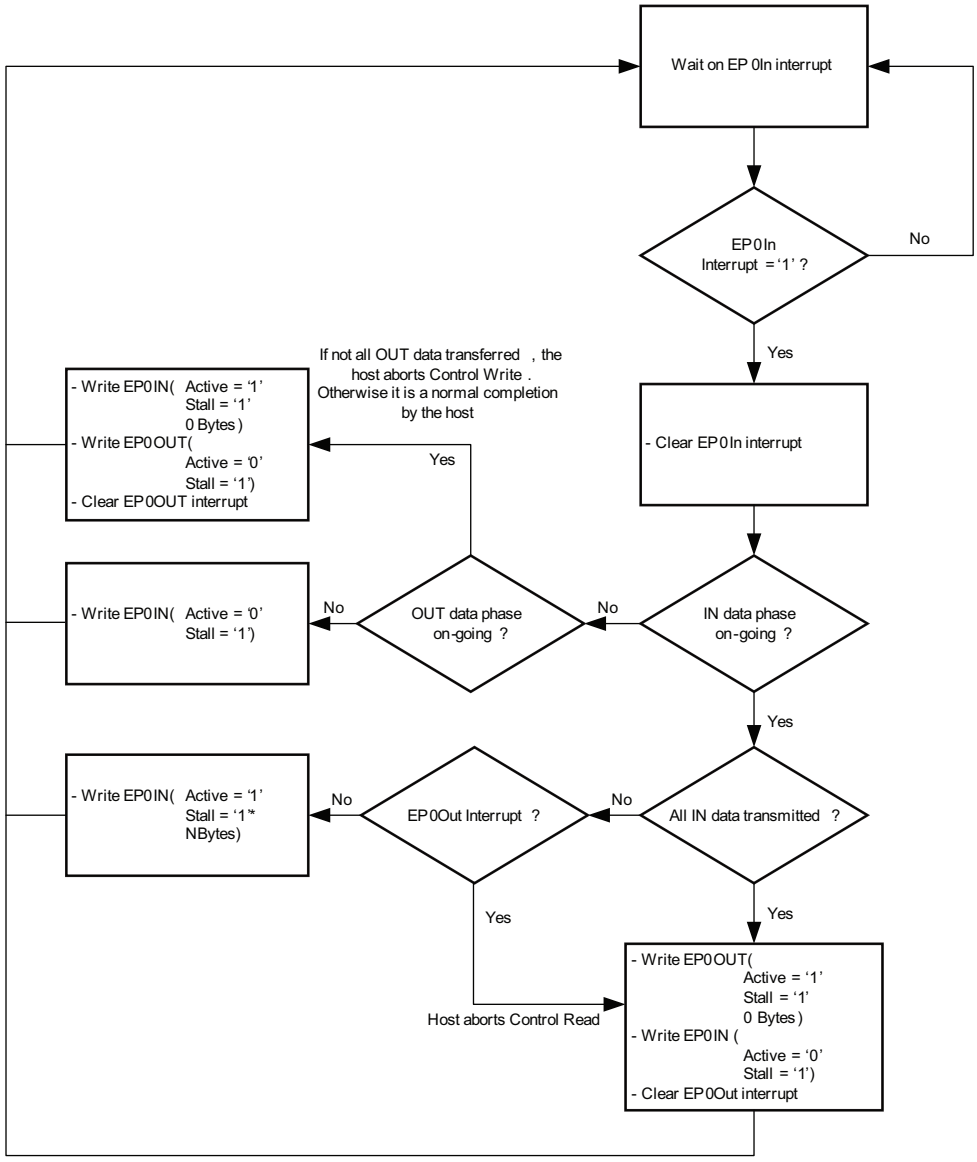


Fig 155. Flowchart of control endpoint 0 - OUT direction



* : STALL bit must only be set when it is the last packet during the data phase for this Control Transfer

Fig 156. Flowchart of control endpoint 0 - IN direction

41.8.3 Generic endpoint: single buffering

To enable single buffering, software must set the corresponding "BUF_SB bit in the "USB EP Buffer Configuration *register* to 0. In the *USB EP Buffer in use* register, the 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 list entry for the desired endpoint and sets the active bit. The hardware will transmit/receive multiple packets for this endpoint until the NBytes value is equal to 0. When NBytes goes to 0, hardware clears the active bit and sets the corresponding endpoint interrupt status bit in INTSTAT.

Software must wait until hardware has cleared the active bit to change the command/status bits in the endpoint command/status list entry. It prevents hardware from overwriting a new value programmed by software with 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 (EPSKIP).

41.8.4 Generic endpoint: double buffering

To enable double buffering, the software must set the corresponding *USB EP Buffer Config* bit to 1. The *USB EP Buffer in use* register indicates which buffer will be used by hardware when the next token is received.

When hardware clears the active bit of the current buffer in use, it will switch the buffer in use. Software can also force hardware to use a certain buffer by writing to the corresponding *USB EP Buffer in use* bit.

41.8.5 Special cases

41.8.5.1 Use of the active bit

The use of the active bit is slightly different between OUT and IN endpoints.

When data must be received for the OUT endpoint, the software will set the active bit to 1 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 1 and programs the NBytes field to the number of bytes that must be transmitted.

41.8.5.2 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 1.

41.8.5.3 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 un-stalled and the toggle bit for that endpoint must be reset to 0. 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.

41.8.5.4 Set configuration

When a SetConfiguration request is received with a configuration value different from 0, 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 41.8.5.3 “Clear feature (endpoint halt)” 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 1.

41.8.6 USB0 wake-up

41.8.6.1 Waking up from deep-sleep mode on USB activity

To allow the chip to wake up from deep-sleep mode on USB activity, complete the following steps:

1. Set bit FORCE_NEEDCLK in the DEVCMDSTAT register, see Section 41.7.1 “USB0 device command/status register” to 0 (default) to enable automatic control of the DEV_NEEDCLK signal.
2. Wait until USB device is suspended by polling the DSUS bit in the DEVCMDSTAT register (DSUS = 1).
3. The DEV_NEEDCLK signal will be de-asserted after another 2 ms. Poll the USB0NEEDCLKSTAT register until the DEV_NEEDCLK status bit is 0. See Section 4.5.63 “USB0 need clock status register”.
4. Clear pending USB0_NEEDCLK activity/wake-up interrupt before enabling it. Enable USB0_NEEDCLK in the NVIC. See Section 3.4.1 “Interrupt set-enable register 0”.
5. Set POL_FS_DEV_NEEDCLK in the USB0CLKCTRL register to trigger the USB activity wake-up interrupt on the rising edge of the DEV_NEEDCLK signal.
6. Enable the wake-up from deep-sleep mode on the USB activity interrupt via the POWER_EnterDeepSleep() low power API.

7. Enter deep-sleep mode via the power API, see Section 14.4.3 “POWER_EnterDeepSleep”.

The chip will automatically wake up and resume execution on USB activity.

41.8.6.2 Remote wake-up

To issue a remote wake-up when the USB activity is suspended, complete the following steps:

1. Set bit FORCE_NEEDCLK in the DEVCMDSTAT register to 0 (Section 41.7.1 “USB0 device command/status register”, default) to enable automatic control of the DEV_NEEDCLK 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 FORCE_NEEDCLK (Section 41.7.1 “USB0 device command/status register”) in the DEVCMDSTAT register.
4. Write a 0 to the DSUS bit in the DSVCMDSTAT register.
5. Wait until the USB leaves the suspend state by polling the DSUS bit in the DSVCMDSTAT register (DSUS =0).
6. Clear the FORCE_NEEDCLK bit (Section 41.7.1 “USB0 device command/status register”, bit 0) in the DEVCMDSTAT to enable automatic USB clock control.

42.1 How to read this chapter

The USB full-speed controller is available on all LPC55S6x/LPC55S2x/LPC552x devices. This chapter describes the host functionality of the controller.

42.2 Introduction

This section describes the host portion of the USB0 full-speed controller USB 2.0.

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

The host controller enables data exchange with various USB devices attached to the bus. It consists of register interface, serial interface engine, and DMA controller. The register interface complies to the OHCI specification.

Table 779. USB (OHCI) related acronyms and abbreviations

Acronym/abbreviation	Description
AHB	Advanced High-Performance Bus
DMA	Direct Memory Access
FS	Full Speed
LS	Low Speed
OHCI/OpenHCI	Open Host Controller Interface
USB	Universal Serial Bus

42.3 Features

- OHCI compliant.
- OpenHCI specifies the operation and interface of the USB host controller and SW driver.
- The host controller has four USB states visible to the SW driver:
  - USBOperational: Process lists and generate SOF tokens.
  - USBReset: Forces reset signaling on the bus, SOF disabled.
  - USBSuspend: Monitor USB for wake-up activity.
  - USBResume: Forces resume signaling on the bus.
- HCCA register points to interrupt and isochronous descriptors list.
- ControlHeadED and BulkHeadED registers point to control and bulk descriptors list.

42.4 Architecture

The architecture of the USB host controller is shown below in Figure 157.

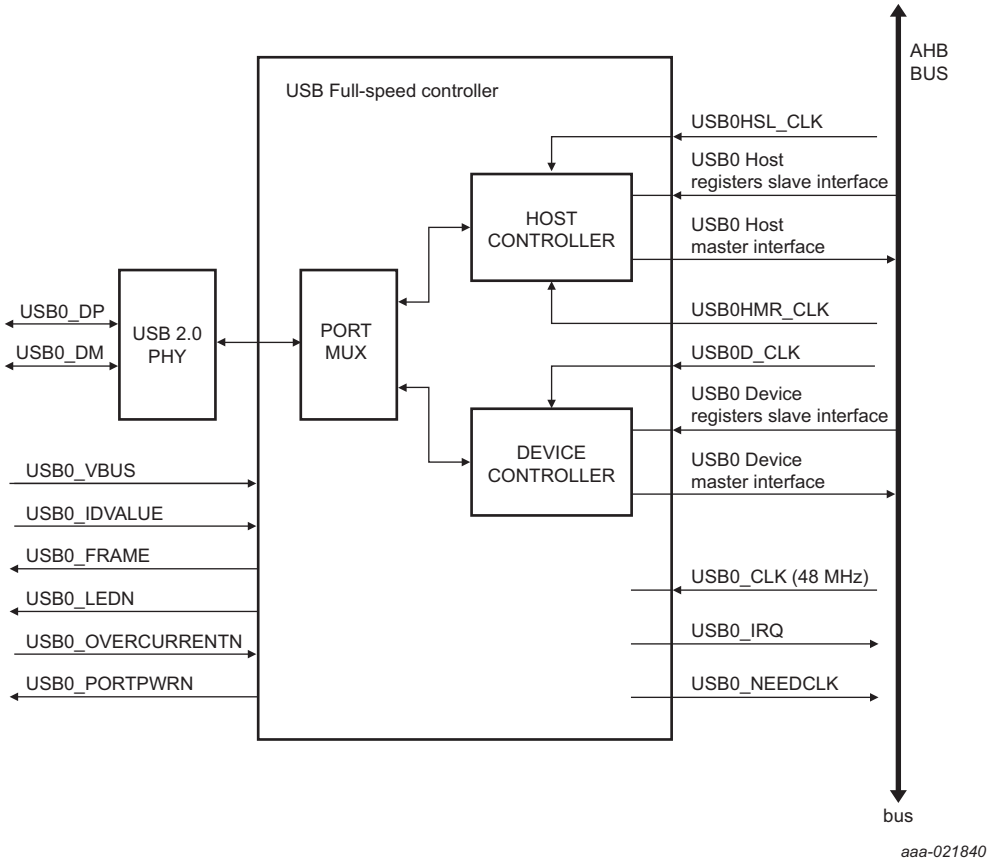


Fig 157. USB full-speed host/device controller block diagram

42.5 Basic configuration

The USB controller is configured using the following registers:

- Power: In the PDRUNCFG0 register, see Table 311, set bit PDEN_USB0_PHY.

  Remark: On reset, the USB block is disabled (PDEN_USB0_PHY = 1).
- Clock: To have the full-speed USB operating, select either the PLL1, or USB PLL, or FRO clock output as the USB0 clock and the clock must be 48 MHz. The CPU clock must be configured to a minimum frequency of 12 MHz. See Section 4.5.40 “USB0 clock source select register” and Section 4.5.55 “USB0 full-speed clock divider register” for more details.
  In AHBCLKCTRL2, set the USB0_HOSTS and USB0_HOSTM bits. See Section 4.5.19 “AHB clock control 2” for more details.
- Port control: Clear DEV_ENABLE bit in PORTMODE register, see Section 42.7.23 “PortMode register” to ensure that the port is controlled by the USB0 host block. Set ID_EN to enable ID pin pull-up.

- Pins: See Table 780 for more details.
- Reset: The USB0 host AHB master and slave can be reset by the USB0_HOSTM_RST and USB0_HOSTS_RST bits in PRESETCTRL2. See Section 4.5.9 “Peripheral reset control 2” for more details.
- Wake-up: Activity on the USB bus port can wake up the microcontroller from deep-sleep mode. See Section 42.6.2.1 “USB0 host wake-up”.
- Interrupts: The USB0 host controller has two interrupt sources allocated in the NVIC interrupt source table: a general interrupt, USB0, and an activity interrupt, USB0_NEEDCLK. See Section 3.3.1 “Interrupt sources” for more details.

42.6 Interfaces

42.6.1 Pin description

Table 780. USB host pin description

Pin name	Port pin	IOCON function, Mode	Direction	Description
USB0_DP	-	-	I/O	Positive differential data.
USB0_DM	-	-	I/O	Negative differential data.
USB0_IDVALUE	PIO0_26	PIO0_26, function 7 Mode: pull-up	I	A-device (host role) or B-device (peripheral role) indication.
USB0_PORTPWRN	PIO1_3 PIO1_12	PIO1_3, function 7 PIO1_12, function 4 Mode: inactive	O	VBUS drive signal (towards external charge pump or power management unit).
USB0_OVERCURRENTN	PIO0_28 PIO1_13	PIO0_28, function 7 PIO1_13, function 4 Mode: inactive	I	Port power fault signal indicating over-current condition; this signal monitors over-current on the USB bus (external circuitry is required to detect over-current condition).
USB0_VBUS	PIO0_22 PIO1_11	PIO0_22, function 7/ Mode: inactive PIO1_11, function 4/ Mode: inactive	I	USB VBUS status input. When this function is not enabled via its corresponding IOCON register, it is driven LOW internally. **Remark:** Enable this pin for correct host operation. For example, it is required detect a change in the connection status.
USB0_FRAME	-	-	-	Device only function.
USB0_LEDN	-	-	-	Device only function.

42.6.2 Software interface

The software interface of the USB host block consists of a register view and the format definitions for the endpoint descriptors. See the OHCI specification for details on these two aspects. Section 42.7 “Register description” shows the register map.

42.6.2.1 USB0 host wake-up

To allow the chip to wake up from deep-sleep mode on USB activity, complete the following steps:

1. Send GET_STATUS command to the connected device and check if it is capable of REMOTE_WAKEUP. See "Get Status" command section in "USB Device Framework" in the USB 2.0 Specification.

2. Check HCRHPORTSTATUS (offset 0x54) register to see if the device is connected. Set PSS to 1 to suspend the port and set HCFS bits to 11b in HCControl register (offset 0x04) to put the host controller into SUSPEND state and then set 3 ms delay for the device to suspend.
3. Set DRWE bit in HCRHStatus (offset 0x50) register to enable remote wake-up.
4. Poll the USB0NEEDCLKSTAT register until the HOST_NEED_CLKST bit is 0, see Section 4.5.63 “USB0 need clock status register”.
5. Set POL_FS_HOST_CLK bit in the USB0CLKCTRL register to 1 to trigger the USB activity wake-up interrupt (USB0_NEEDCLK) on the rising edge of the USB0 host NEEDCLK signal, see Section 4.5.62 “USB0 need clock control register”.
6. Enable the wake-up from deep-sleep mode on the USB activity interrupt via the POWER_EnterDeepSleep() low power API.
7. Clear pending USB0 activity interrupt, USB0_NEEDCLK, see Section 3.3.1 “Interrupt sources” before enabling it.
8. Enter deep-sleep mode via power API, see Section 14.4.3 “POWER_EnterDeepSleep”. The chip will automatically wake-up and resume execution on USB activity.
9. Re-initialize the USB host controller after the USB0_NEEDCLK interrupt is invoked.

42.7 Register description

The following registers are located in the AHB clock domain. They can be accessed directly by the processor. All registers are 32 bits wide and aligned with word address boundaries.

Table 781. Register overview: USB host register address definitions (base address 0x400A 2000)

Name	Access	Offset	Description	Reset value	Section
HCREVISION	RO	0x00	BCD representation of the version of the HCI specification that is implemented by the Host Controller (HC).	0x10	42.7.1
HCCONTROL	R/W	0x04	Defines the operating modes of the HC.	0x0	42.7.2
HCCOMMANDSTATUS	R/W	0x08	This register is used to receive the commands from the Host Controller Driver (HCD). It also indicates the status of the HC.	0x0	42.7.3
HCINTERRUPTSTATUS	R/W	0x0C	Indicates the status on various events that cause hardware interrupts by setting the appropriate bits.	0x0	42.7.4
HCINTERRUPTENABLE	R/W	0x10	Controls the bits in the HcInterruptStatus register and indicates which events will generate a hardware interrupt.	0x0	42.7.5
HCINTERRUPTDISABLE	R/W	0x14	The bits in this register are used to disable corresponding bits in the HCInterruptStatus register and in turn disable that event leading to hardware interrupt.	0x0	42.7.6
HCHCCA	R/W	0x18	Contains the physical address of the host controller communication area.	0x0	42.7.7
HCPERIODCURRENTED	R	0x1C	Contains the physical address of the current isochronous or interrupt endpoint descriptor.	0x0	42.7.8
HCCONTROLHEADED	R/W	0x20	Contains the physical address of the first endpoint descriptor of the control list.	0x0	42.7.9
HCCONTROLCURRENTED	R/W	0x24	Contains the physical address of the current endpoint descriptor of the control list	0x0	42.7.10
HCBULKHEADED	R/W	0x28	Contains the physical address of the first endpoint descriptor of the bulk list.	0x0	42.7.11
HCBULKCURRENTED	R/W	0x2C	Contains the physical address of the current endpoint descriptor of the bulk list.	0x0	42.7.12
HCDONEHEAD	R	0x30	Contains the physical address of the last transfer descriptor added to the ‘Done’ queue.	0x0	42.7.13
HCFMINTERVAL	R/W	0x34	Defines the bit time interval in a frame and the full speed maximum packet size which would not cause an overrun.	0x2EDF	42.7.14
HCFMREMAINING	R	0x38	A 14-bit counter showing the bit time remaining in the current frame.	0x0	42.7.15
HCFMNUMBER	R	0x3C	Contains a 16-bit counter and provides the timing reference among events happening in the HC and the HCD.	0x0	42.7.16
HCPERIODICSTART	R/W	0x40	Contains a programmable 14-bit value which determines the earliest time HC should start processing a periodic list.	0x0	42.7.17

Table 781. Register overview: USB host register address definitions (base address 0x400A 2000) *…continued*

Name	Access	Offset	Description	Reset value	Section
HCLSTHRESHOLD	R/W	0x44	Contains 11-bit value which is used by the HC to determine whether to commit to transfer a maximum of 8-byte LS packet before EOF.	0x628	42.7.18
HCRHDESCRIPTORA	R/W	0x48	First of the two registers which describes the characteristics of the root hub.	0xFF000902	42.7.19
HCRHDESCRIPTORB	R/W	0x4C	Second of the two registers which describes the characteristics of the Root Hub.	0x60000	42.7.20
HCRHSTATUS	R/W	0x50	This register is divided into two parts. The lower D-word represents the hub status field and the upper word represents the hub status change field.	0x0	42.7.21
HCRHPORTSTATUS	R/W	0x54	Controls and reports the port events on a per-port basis.	0x0	42.7.22
PORTMODE	R/W	0x5C	Controls the port if it is attached to the host block or the device block.	0x0	42.7.23

42.7.1 Host controller revision register

Table 782. Host controller revision register (HCREVISION, offset 0x00)

Bit	Symbol	Description	Reset value
7:0	REV	Revision. This read-only field contains the BCD representation of the version of the HCI specification that is implemented by this HC.	0x10
31:8	-	Reserved	-

42.7.2 Host controller control register

Table 783. Host controller control register (HCCONTROL, offset 0x04)

Bit	Symbol	Description	Reset value
1:0	CBSR	ControlBulkServiceRatio. This specifies the service ratio between control and bulk EDs. Before processing any of the nonperiodic lists, HC must compare the ratio specified with its internal count on how many nonempty control EDs have been processed, in determining whether to continue serving another control ED or switching to bulk EDs. The internal count will be retained when crossing the frame boundary. In case of reset, HCD is responsible for restoring this value.	0x0
	CBSR Value	**Number of control EDs Over bulk EDs Served**	
	0	1:1	
	1	2:1	
	2	3:1	
	3	4:1	
2	PLE	PeriodicListEnable. This bit is set to enable the processing of the periodic list in the next frame. If cleared by HCD, processing of the periodic list does not occur after the next SOF. HC must check this bit before it starts processing the list.	0
3	IE	IsochronousEnable. This bit is used by HCD to enable/disable processing of isochronous EDs. While processing the periodic list in a frame, HC checks the status of this bit when it finds an Isochronous ED (F=1). If set (enabled), HC continues processing the EDs. If cleared (disabled), HC halts processing of the periodic list (that contains only isochronous EDs) and begins processing the bulk/control lists. Setting this bit is guaranteed to take effect in the next frame (not the current frame).	0
4	CLE	ControlListEnable. This bit is set to enable the processing of the control list in the next frame. If cleared by HCD, processing of the control list does not occur after the next SOF. HC must check this bit whenever it determines to process the list. When disabled, HCD may modify the list. If HcControlCurrentED is pointing to an ED to be removed, HCD must advance the pointer by updating HcControlCurrentED before re-enabling processing of the list.	0
5	BLE	BulkListEnable This bit is set to enable the processing of the bulk list in the next frame. If cleared by HCD, processing of the bulk list does not occur after the next SOF. HC checks this bit whenever it determines to process the list. When disabled, HCD may modify the list. If HcBulkCurrentED is pointing to an ED to be removed, HCD must advance the pointer by updating HcBulkCurrentED before re-enabling processing of the list.	0

Table 783. Host controller control register (HCCONTROL, offset 0x04) *…continued*

Bit	Symbol	Description	Reset value
7:6	HCFS	HostControllerFunctionalState for USB 00b: USBRESET 01b: USBRESUME 10b: USBOPERATIONAL 11b: USBSUSPEND A transition to USBOPERATIONAL from another state causes SOFgeneration to begin 1 ms later. HCD may determine whether HC has begun sending SOFs by reading the StartofFrame field of HcInterruptStatus. This field may be changed by HC only when in the USBSUSPEND state. HC may move from the USBSUSPEND state to the USBRESUME state after detecting the resume signaling from a downstream port. HC enters USBSUSPEND after a software reset, whereas it enters USBRESET after a hardware reset. The latter also resets the root hub and asserts subsequent reset signaling to downstream ports.	0x0
8	IR	InterruptRouting This bit determines the routing of interrupts generated by events registered in HcInterruptStatus. If clear, all interrupts are routed to the normal host bus interrupt mechanism. If set, interrupts are routed to the system management interrupt. HCD clears this bit upon a hardware reset, but it does not alter this bit upon a software reset. HCD uses this bit as a tag to indicate the ownership of HC.	0
9	RWC	RemoteWakeupConnected This bit indicates whether HC supports remote wake-up signaling. If remote wake-up is supported and used by the system it is the responsibility of system firmware to set this bit during POST. HC clears the bit upon a hardware reset but does not alter it upon a software reset. Remote wake-up signaling of the host system is host-bus-specific and is not described in this specification.	0
10	RWE	RemoteWakeupEnable This bit is used by HCD to enable or disable the remote wake-up feature upon the detection of upstream resume signaling. When this bit is set and the ResumeDetected bit in HcInterruptStatus is set, a remote wake-up is signaled to the host system. Setting this bit has no impact on the generation of hardware interrupt.	0
31:11	-	Reserved.	-

42.7.3 Host controller command status register

Table 784. Host controller command status register (HCCOMMANDSTATUS, offset 0x08)

Bit	Symbol	Description	Reset value
0	HCR	HostControllerReset This bit is set by HCD to initiate a software reset of HC. Regardless of the functional state of HC, it moves to the USBSUSPEND state in which most of the operational registers are reset except those stated otherwise; For example, the InterruptRouting field of HcControl and no host bus accesses are allowed. This bit is cleared by HC when the reset operation is completed. The reset operation must be completed within 10 μs. This bit, when set, should not cause a reset to the Root Hub and no subsequent reset signaling should be asserted to its downstream ports.	0
1	CLF	ControlListFilled This bit is used to indicate whether there are any TDs on the control list. It is set by HCD whenever it adds a TD to an ED in the control list. When HC begins to process the head of the control list, it checks CLF. As long as ControlListFilled is 0, HC will not start processing the control list. If CF is 1, HC will start processing the control list and will set ControlListFilled to 0. If HC finds a TD on the list, then HC will set ControlListFilled to 1 causing the control list processing to continue. If no TD is found on the control list, and if the HCD does not set ControlListFilled, then ControlListFilled will still be 0 when HC completes processing the control list and control list processing will stop.	0
2	BLF	BulkListFilled This bit is used to indicate whether there are any TDs on the bulk list. It is set by HCD whenever it adds a TD to an ED in the bulk list. When HC begins to process the head of the bulk list, it checks BF. As long as BulkListFilled is 0, HC will not start processing the bulk list. If BulkListFilled is 1, HC will start processing the bulk list and will set BF to 0. If HC finds a TD on the list, then HC will set BulkListFilled to 1 causing the bulk list processing to continue. If no TD is found on the bulk list, and if HCD does not set BulkListFilled, then BulkListFilled will still be 0 when HC completes processing the bulk list and bulk list processing will stop.	0
3	OCR	OwnershipChangeRequest This bit is set by an OS HCD to request a change of control of the HC. When set, HC will set the OwnershipChange field in HcInterruptStatus. After the change over, this bit is cleared and remains so until the next request from OS HCD.	0
5:4	-	Reserved	-
7:6	SOC	SchedulingOverrunCount These bits are incremented on each scheduling overrun error. It is initialized to 00b and wraps around at 11b. This will be incremented when a scheduling overrun is detected even if SchedulingOverrun in HcInterruptStatus has already been set. It is used by HCD to monitor any persistent scheduling problems.	0
31:8	-	Reserved	-

42.7.4 Host controller interrupt status register

The HC interrupt status register provides status on various events that cause hardware interrupts. When an event occurs, host controller sets the corresponding bit in this register. When a bit becomes set, a hardware interrupt is generated if the interrupt is

enabled in the HcInterruptEnable register (see Section 42.7.5) and the MasterInterruptEnable bit is set. The host controller driver may clear specific bits in this register by writing 1 to bit positions to be cleared. The host controller driver may not set any of these bits. The host controller will never clear the bit.

Table 785. Host controller interrupt status register (HCINTERRUPTSTATUS, offset 0x0C)

Bit	Symbol	Description	Reset value
0	SO	SchedulingOverrun This bit is set when the USB schedule for the current frame overruns and after the update of HccaFrameNumber. A scheduling overrun will also cause the SchedulingOverrunCount of HcCommandStatus to be incremented.	0
1	WDH	WritebackDoneHead This bit is set immediately after HC has written HcDoneHead to HccaDoneHead. Further updates of the HccaDoneHead will not occur until this bit has been cleared. HCD should only clear this bit after it has saved the content of HccaDoneHead.	0
2	SF	StartofFrame This bit is set by HC at each start of a frame and after the update of HccaFrameNumber. HC also generates a SOF token at the same time.	0
3	RD	ResumeDetected This bit is set when HC detects that a device on the USB is asserting resume signaling. It is the transition from no resume signaling to resume signaling causing this bit to be set. This bit is not set when HCD sets the USBRESUME state.	0
4	UE	UnrecoverableError This bit is set when HC detects a system error not related to USB. HC should not proceed with any processing nor signaling before the system error has been corrected. HCD clears this bit after HC has been reset.	0
5	FNO	FrameNumberOverflow This bit is set when the MSb of HcFmNumber (bit 15) changes value, from 0 to 1 or from 1 to 0, and after HccaFrameNumber has been updated	0
6	RHSC	RootHubStatusChange This bit is set when the content of HcRhStatus or the content of any of HcRhPortStatus[NumberofDownstreamPort] has changed.	0
9:7	-	Reserved	-
31:10	OC	OwnershipChange This bit is set by HC when HCD sets the OwnershipChangeRequest field in HcCommandStatus. This event, when unmasked, will always generate an System Management Interrupt (SMI) immediately. This bit is tied to 0b when the SMI pin is not implemented.	0

42.7.5 Host controller interrupt enable register

In the HcInterruptEnable register, each enable bit corresponds to an associated interrupt bit in the HcInterruptStatus register. The HcInterruptEnable register is used to control which events generate a hardware interrupt. When a bit is set in the HcInterruptStatus register AND the corresponding bit in the HcInterruptEnable register is set AND the MasterInterruptEnable bit is set, then a hardware interrupt is requested on the host bus.

Writing a '1' to a bit in this register sets the corresponding bit, whereas writing a '0' to a bit in this register leaves the corresponding bit unchanged. On read, the current value of this register is returned.

Table 786. Host controller interrupt enable register (HCINTERRUPTENABLE, offset 0x10)

Bit	Symbol	Value	Description	Reset value
0	SO		Scheduling Overrun interrupt.	0
		0	No effect.	
		1	Enables interrupt.	
1	WDH		HcDoneHead Writeback interrupt.	0
		0	No effect.	
		1	Enables interrupt.	
2	SF		Start of frame interrupt.	0
		0	No effect.	
		1	Enables interrupt.	
3	RD		Resume Detect interrupt.	0
		0	No effect.	
		1	Enables interrupt.	
4	UE		Unrecoverable Error interrupt.	0
		0	No effect.	
		1	Enables interrupt.	
5	FNO		Frame Number Overflow interrupt.	0
		0	No effect.	
		1	Enables interrupt.	
6	RHSC		Root Hub Status Change interrupt.	0
		0	No effect.	
		1	Enables interrupt.	
29:7	-		Reserved.	-
30	OC		Ownership Change interrupt.	
		0	No effect.	
		1	Enables interrupt.	
31	MIE		Master Interrupt Enable. It is used by HCD as a master interrupt enable. A 0 written to this field is ignored by HC. A 1 written to this field enables interrupt generation because of events specified in the other bits of this register.	0

42.7.6 Host controller interrupt disable register

Each disable bit in the HcInterruptDisable register corresponds to an associated interrupt bit in the HcInterruptStatus register. The HcInterruptDisable register is coupled with the HcInterruptEnable register. Therefore, writing a 1 to a bit in this register clears the corresponding bit in the HcInterruptEnable register and writing a 0 to a bit in this register leaves the corresponding bit in the HcInterruptEnable register unchanged. On read, the current value of the HcInterruptEnable register is returned.

Table 787. Host controller interrupt disable register (HCINTERRUPTDISABLE, offset 0x14)

Bit	Symbol	Value	Description	Reset value
0	SO		Scheduling Overrun interrupt.	0
		0	No effect.	
		1	Disables interrupt.	
1	WDH		HcDoneHead Writeback interrupt.	0
		0	No effect.	
		1	Disables interrupt.	
2	SF		Start of frame interrupt.	0
		0	No effect.	
		1	Disables interrupt.	
3	RD		Resume Detect interrupt.	0
		0	No effect.	
		1	Disables interrupt.	
4	UE		Unrecoverable Error interrupt.	0
		0	No effect.	
		1	Disables interrupt.	
5	FNO		Frame Number Overflow interrupt.	0
		0	No effect.	
		1	Disables interrupt.	
6	RHSC		Root Hub Status Change interrupt.	0
		0	No effect.	
		1	Disables interrupt.	
29:7	-		Reserved.	-
30	OC		Ownership Change interrupt.	
		0	No effect.	
		1	Disables interrupt.	
31	MIE		A 0 written to this field is ignored by HC. A 1 written to this field disables interrupt generation due to events specified in the other bits of this register. This field is set after a hardware or software reset.	0

42.7.7 Host controller communication area register

Table 788. Host controller communication area register (HCHCCA, offset 0x18)

Bit	Symbol	Description	Reset value
7:0	-	Reserved.	0
31:8	HCCA	Base address of the host controller communication area.	-

42.7.8 Host controller period current ED register

The host controller period current ED register is used by the host controller to point to the head of one of the Periodic lists, which will be processed in the current frame.

Table 789. Host controller period current ED register (HCPERIODCURRENTED, offset 0x1C)

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	-
31:4	PCED	The content of this register is updated by HC after a periodic ED is processed. HCD may read the content in determining which ED is currently being processed at the time of reading.	0

42.7.9 Host controller control head ED register

The host controller control head ED register contains the physical address of the first Endpoint Descriptor of the control list.

Table 790. Host controller control head ED register (HCCONTROLHEADED, offset 0x20)

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	-
31:4	CHED	HC traverses the control list starting with the HcControlHeadED pointer. The content is loaded from HCCA during the initialization of HC.	0

42.7.10 Host controller control current ED register

Table 791. Host controller control current ED register (HCCONTROLCURRENTED, offset 0x24)

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	-
31:4	CCED	ControlCurrentED. This pointer is advanced to the next ED after serving the current one. HC continues to process the list from where it left off in the last frame. When it reaches the end of the control list, HC checks the ControlListFilled (CLF) bit in the HcCommandStatus. If set, it copies the content of HcControlHeadED to HcControlCurrentED and clears the bit. If not set, it does nothing. HCD is allowed to modify this register only when the ControlListEnable of HcControl is cleared. When set, HCD only reads the instantaneous value of this register. Initially, it is set to 0 to indicate the end of the control list.	0

42.7.11 Host controller bulk head ED register

Table 792. Host controller bulk head ED register (HCBULKHEADED, offset 0x28)

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	-
31:4	BHED	BulkHeadED HC traverses the bulk list starting with the HcBulkHeadED pointer. The content is loaded from HCCA during the initialization of HC.	0

42.7.12 Host controller bulk current ED register

Table 793. Host controller bulk current ED register (HCBULKCURRENTED, offset 0x2C)

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	-
31:4	BCED	BulkCurrentED It is advanced to the next ED after the HC has served the current one. HC continues to process the list from where it left off in the last frame. When it reaches the end of the bulk list, HC checks the ControlListFilled of HcControl. If set, it copies the content of HcBulkHeadED to HcBulkCurrentED and clears the bit. If it is not set, it does nothing. HCD is only allowed to modify this register when the BulkListEnable of HcControl is cleared. When set, the HCD only reads the instantaneous value of this register. It is initially set to 0 to indicate the end of the bulk list.	0

42.7.13 Host controller done head register

Table 794. Host controller done head register (HCDONEHEAD, offset 0x30)

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	-
31:4	DH	DoneHead When a TD is completed, HC writes the content of HcDoneHead to the NextTD field of the TD. HC then overwrites the content of HcDoneHead with the address of this TD. It is set to 0 whenever HC writes the content of this register to HCCA. It also sets the WritebackDoneHead of HcInterruptStatus.	0

42.7.14 Host controller frame interval register

Table 795. Host controller frame interval register (HCFMINTERVAL, offset 0x34)

Bit	Symbol	Description	Reset value
13:0	FI	FrameInterval This specifies the interval between two consecutive SOFs in bit times. The nominal value is set to be 11,999. HCD should store the current value of this field before resetting HC. By setting the HostControllerReset field of HcCommandStatus, the HC resets this field to its nominal value. HCD may choose to restore the stored value on completion of the Reset sequence.	0x2EDF
15:14	-	Reserved	-
30:16	FSMPS	FSLargestDataPacket This field specifies a value which is loaded into the largest data packet counter at the beginning of each frame. The counter value represents the largest amount of data in bits which can be sent or received by the HC in a single transaction at any given time without causing scheduling overrun. The field value is calculated by the HCD.	-
31	FIT	FrameIntervalToggle HCD toggles this bit whenever it loads a new value to FrameInterval.	0

42.7.15 Host controller frame remaining register

The host controller frame remaining register is a 14-bit down counter showing the bit time remaining in the current frame.

Table 796. Host controller frame remaining register (HCFMREMAINING, offset 0x38)

Bit	Symbol	Description	Reset value
13:0	FR	FrameRemaining This counter is decremented at each bit time. When it reaches 0, it is reset by loading the FrameInterval value specified in HcFmInterval at the next bit time boundary. When entering the USBOPERATIONAL state, HC re-loads the content with the FrameInterval of HcFmInterval and uses the updated value from the next SOF.	2EDFh
30:14	-	Reserved	-
31	FRT	FrameRemainingToggle This bit is loaded from the FrameIntervalToggle field of HcFmInterval whenever FrameRemaining reaches 0. This bit is used by HCD for the synchronization between FrameInterval and FrameRemaining.	0

42.7.16 Host controller frame number register

The host controller frame number register is a 16-bit counter. It provides a timing reference among events happening in the host controller and the host controller driver. The host controller driver may use the 16-bit value specified in this register and generate a 32-bit frame number without requiring frequent access to the register.

Table 797. Host controller frame number register (HCFMNUMBER, offset 0x3C)

Bit	Symbol	Description	Reset value
15:0	FN	FrameNumber It is incremented when HcFmRemaining is re-loaded. It will be rolled over to 0h after FFFH. When entering the USBOPERATIONAL state, it is incremented automatically. The content is written to HCCA after HC has incremented the FrameNumber at each frame boundary and sent a SOF but before HC reads the first ED in that frame. After writing to HCCA, HC sets the StartofFrame in HcInterruptStatus.	0
31:16	-	Reserved.	-

42.7.17 Host controller periodic start register

The host controller periodic start register has a 14-bit programmable value that determines the earliest time when HC should start processing the periodic list.

Table 798. Host controller periodic start register (HCPERIODICSTART, offset 0x40)

Bit	Symbol	Description	Reset value
13:0	PS	PeriodicStart After a hardware reset, this field is cleared and then set by HCD during the HC initialization. The value is calculated approximately as 10% off from HcFmInterval.. A typical value will be 3E67h. When HcFmRemaining reaches the value specified, processing of the periodic lists will have priority over control/bulk processing. HC will therefore start processing the interrupt list after completing the current control or bulk transaction that is in progress.	0
31:11	-	Reserved.	-

42.7.18 Host controller LS threshold register

The host controller LS threshold register contains an 11-bit value used by the host controller to determine whether to commit to the transfer of a maximum of 8-byte LS packet before EOF. The host controller and the host controller driver are not allowed to change this value.

Table 799. Host controller LS threshold register (HCLSTHRESHOLD, offset 0x44)

Bit	Symbol	Description	Reset value
11:0	LST	LSThreshold This field contains a value which is compared to the FrameRemaining field prior to initiating a low-speed transaction. The transaction is started only if FrameRemaining ≥ this field. The value is calculated by HCD with the consideration of transmission and setup overhead.	0x628
31:12	-	Reserved.	-

42.7.19 Host controller root hub descriptor A register

The host controller root hub descriptor A register is the first register describing the characteristics of the root hub. Reset values are implementation specific. The descriptor length (11), descriptor type, and hub controller current (0) fields of the hub Class Descriptor are emulated by the HCD. All other fields are located in the HcRhDescriptorA and HcRhDescriptorB registers.

Table 800. Host controller root hub descriptor register (HCRHDESCRIPTORA offset 0x48)

Bit	Symbol	Description	Reset value
7:0	NDP	NumberDownstreamPorts These bits specify the number of downstream ports supported by the root hub. It is implementation-specific. The minimum number of ports is 1. The maximum number of ports supported by OpenHCI is 15.	0x2
8	PSM	PowerSwitchingMode This bit is used to specify how the power switching of the root hub ports is controlled. It is implementation-specific. This field is only valid if the NoPowerSwitching field is cleared. 0: All ports are powered at the same time. 1: Each port is powered individually. This mode allows portpower to be controlled by either the global switch or per-port switching. If the PortPowerControlMask bit is set, the port responds only to port power commands (Set/ClearPortPower). If the port mask is cleared, the port is controlled only by the global power switch (Set/ClearGlobalPower).	1
9	NPS	NoPowerSwitching These bits are used to specify whether power switching is supported or port are always powered. It is implementationspecific. When this bit is cleared, the PowerSwitchingMode specifies global or per-port switching. 0: Ports are power switched. 1: Ports are always powered on when the HC is powered on.	0
10	DT	DeviceType This bit specifies that the root hub is not a compound device. The root hub is not permitted to be a compound device. This field should always read/write 0.	0
11	OCPM	OverCurrentProtectionMode This bit describes how the overcurrent status for the root hub ports are reported. At reset, this fields should reflect the same mode as PowerSwitchingMode. This field is valid only if the NoOverCurrentProtection field is cleared. 0: Over-current status is reported collectively for all downstream ports. 1: Over-current status is reported on a per-port basis.	1
12	NOCP	NoOverCurrentProtection This bit describes how the overcurrent status for the root hub ports are reported. When this bit is cleared, the OverCurrentProtectionMode field specifies global or per-port reporting. 0: Over-current status is reported collectively for all downstream ports. 1: No overcurrent protection supported.	0
23:13	-	Reserved	-
31:24	POTPGT	PowerOnToPowerGoodTime This byte specifies the duration the HCD has to wait before accessing a powered-on port of the root hub. It is implementation-specific. The unit of time is 2 ms. The duration is calculated as POTPGT * 2 ms.	0xF

42.7.20 Host controller root hub descriptor B register

The host controller root hub descriptor B register is the second register describing the characteristics of the root hub. These fields are written during initialization to correspond with the system implementation. Reset values are implementation-specific.

Table 801. Host controller root hub descriptor register (HCRHDESCRIPTORB offset 0x4C)

Bit	Symbol	Description	Reset value
15:0	DR	DeviceRemovable Each bit is dedicated to a port of the root hub. When cleared, the attached device is removable. When set, the attached device is not removable. bit 0: Reserved bit 1: Device attached to port #1. bit 2: Device attached to port #2. ..... bit 15: Device attached to port #15.	0
31:16	PPCM	PortPowerControlMask Each bit indicates if a port is affected by a global power control command when PowerSwitchingMode is set. When set, the port's power state is only affected by per-port power control (Set/ClearPortPower). When cleared, the port is controlled by the global power switch (Set/ClearGlobalPower). If the device is configured to global switching mode (PowerSwitchingMode = 0), this field is not valid. bit 0: Reserved. bit 1: Ganged-power mask on port #1. bit 2: Ganged-power mask on port #2. ... bit15: Ganged-power mask on port #5.	0

42.7.21 Host controller root hub status register

The host controller root hub status register is divided into two parts. The lower word of a Dword represents the hub status field and the upper word represents the hub status change field. Reserved bits should always be written 0.

Table 802. Host controller root hub status register (HCRHSTATUS register offset 0x50)

Bit	Symbol	Description	Reset value
0	LPS	(Read) LocalPowerStatus The root hub does not support the local power status feature; thus, this bit is always read as 0. (write) ClearGlobalPower In global power mode (PowerSwitchingMode=0), this bit is written to 1 to turn off power to all ports (clear PortPowerStatus). In per-port power mode, it clears PortPowerStatus only on ports whose PortPowerControlMask bit is not set. Writing a 0 has no effect.	0
1	OCI	OverCurrentIndicator This bit reports overcurrent conditions when the global reporting is implemented. When set, an overcurrent condition exists. When cleared, all power operations are normal. If per-port overcurrent protection is implementedthis bit is always 0.	0
14:2	-	Reserved	-

Table 802. Host controller root hub status register (HCRHSTATUS register offset 0x50) *...continued*

Bit	Symbol	Description	Reset value
15	DRWE	(Read) DeviceRemoteWakeupEnable This bit enables a ConnectStatusChange bit as a resume event, causing a USBSUSPEND to USBRESUME state transition and setting the ResumeDetected interrupt. 0 = ConnectStatusChange is not a remote wakeup event. 1 = ConnectStatusChange is a remote wakeup event. (Write) SetRemoteWakeupEnable Writing a '1' sets DeviceRemoveWakeupEnable. Writing a 0 has no effect.	0
16	LPSC	(Read) LocalPowerStatusChange The root hub does not support the local power status feature. Therefore, this bit is always read as 0. (Write) SetGlobalPower In global power mode (PowerSwitchingMode=0), this bit is written to 1 to turn on power to all ports (clear PortPowerStatus). In per-port power mode, it sets PortPowerStatus only on ports whose PortPowerControlMask bit is not set. Writing a 0 has no effect.	0
17	OCIC	OverCurrentIndicatorChange This bit is set by hardware when a change has occurred to the OCI field of this register. The HCD clears this bit by writing a 1. Writing a 0 has no effect.	0
30:18	-	Reserved	-
31	CRWE	(Write) ClearRemoteWakeupEnable Writing a 1 clears DeviceRemoveWakeupEnable. Writing a 0 has no effect.	0

42.7.22 Host controller root hub port status [1:NDP] register

Remark: In LPC55S6x/LPC55S2x/LPC552x, Number Downstream Ports (NDP) = 1

The HcRhPortStatus[1:NDP] register is used to control and report port events on a per-port basis. NumberDownstreamPorts represents the number of HcRhPortStatus registers that are implemented in the hardware. The lower word is used to reflect the port status and the upper word reflects the status change bits. Some status bits are implemented with special write behavior. If a transaction (token through handshake) is in progress when a write to change port status occurs, the resulting port status change must be postponed until the transaction completes. Reserved bits should always be written 0.

Table 803. Host controller root hub port status register (HCRHPORTSTATUS[1:NDP] register offset 0x54)

Bit	Symbol	Description	Reset value
0	CCS	(Read) CurrentConnectStatus This bit reflects the current state of the downstream port. 0 = no device connected. 1 = device connected. (write) ClearPortEnable The HCD writes a 1 to this bit to clear the PortEnableStatus bit. Writing a 0 has no effect. The CurrentConnectStatus is not affected by any write. **Remark:** This bit is always read 1b when the attached device is nonremovable (DeviceRemoveable[NDP]).	0
1	PES	(Read) PortEnableStatus This bit indicates whether the port is enabled or disabled. The root hub may clear this bit when an overcurrent condition, disconnect event, switched-off power, or operational bus error, such as, babble is detected. The change also causes PortEnabledStatusChange to be set. HCD sets this bit by writing SetPortEnable and clears it by writing ClearPortEnable.This bit cannot be set when CurrentConnectStatus is cleared. This bit is also set, if not already, at the completion of a port reset when ResetStatusChange is set or port suspend when SuspendStatusChange is set. 0 = port is disabled. 1 = port is enabled. (Write) SetPortEnable The HCD sets PortEnableStatus by writing a 1. Writing a 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortEnableStatus, but instead sets ConnectStatusChange. This informs the driver that it attempted to enable a disconnected port.	0
2	PSS	(Read) PortSuspendStatus This bit indicates the port is suspended or in the resume sequence. It is set by a SetSuspendState write and cleared when PortSuspendStatusChange is set at the end of the resume interval. This bit cannot be set if CurrentConnectStatus is cleared. This bit is also cleared when PortResetStatusChange is set at the end of the port reset or when the HC is placed in the USBRESUME state. If an upstream resume is in progress, it should propagate to the HC. 0 = port is not suspended. 1 = port is suspended. (Write) SetPortSuspend The HCD sets the PortSuspendStatus bit by writing a 1 to this bit. Writing a 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortSuspendStatus, instead, it sets ConnectStatusChange. This informs the driver that it attempted to suspend a disconnected port.	0

Table 803. Host controller root hub port status register (HCRHPORTSTATUS[1:NDP] register offset 0x54) *…continued*

Bit	Symbol	Description	Reset value
3	POCI	(Read) PortOverCurrentIndicator This bit is only valid when the Root Hub is configured in such a way that overcurrent conditions are reported on a per-port basis. If per-port overcurrent reporting is not supported, this bit is set to 0. If cleared, all power operations are normal for this port. If set, an overcurrent condition exists on this port. This bit always reflects the overcurrent input signal. 0 = no overcurrent condition. 1 = overcurrent condition detected. (write) ClearSuspendStatus The HCD writes a 1 to initiate a resume. Writing a 0 has no effect. A resume is initiated only if PortSuspendStatus is set.	0
4	PRS	(Read) PortResetStatus When this bit is set by a write to SetPortReset, port reset signaling is asserted. When reset is completed, this bit is cleared when PortResetStatusChange is set. This bit cannot be set if CurrentConnectStatus is cleared. 0 = port reset signal is not active. 1 = port reset signal is active. (Write) SetPortReset The HCD sets the port reset signaling by writing a 1 to this bit. Writing a 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortResetStatus, instead, sets ConnectStatusChange. This informs the driver that it attempted to reset a disconnected port.	0
7:5	-	Reserved	-
8	PPS	(Read) PortPowerStatus This bit reflects the port's power status, regardless of the type of power switching implemented. This bit is cleared if an overcurrent condition is detected. HCD sets this bit by writing SetPortPower or SetGlobalPower. HCD clears this bit by writing ClearPortPower or ClearGlobalPower. The PowerSwitchingMode and PortPortControlMask[NDP] determine which power control switches are enabled. In global switching mode (PowerSwitchingMode=0), only Set/ClearGlobalPower controls this bit. In per-port power switching (PowerSwitchingMode=1), if the PortPowerControlMask[NDP] bit for the port is set, only Set/ClearPortPower commands are enabled. If the mask is not set, only Set/ClearGlobalPower commands are enabled. When port power is disabled, CurrentConnectStatus, PortEnableStatus, PortSuspendStatus, and PortResetStatus should be reset. 0 = port power is off. 1 = port power is on. The HCD writes a 1 to set the PortPowerStatus bit. Writing a 0 has no effect.	0

Table 803. Host controller root hub port status register (HCRHPORTSTATUS[1:NDP] register offset 0x54) *…continued*

Bit	Symbol	Description	Reset value
9	LSDA	(Read) LowSpeedDeviceAttached This bit indicates the speed of the device attached to this port. When set, a low-speed device is attached to this port. When clear, a full-speed device is attached to this port. This field is valid only when the CurrentConnectStatus is set. 0 = full speed device attached. 1 = low speed device attached. (write) ClearPortPower The HCD clears the PortPowerStatus bit by writing a 1 to this bit. Writing a 0 has no effect.	0
15:10	-	Reserved	-
16	CSC	ConnectStatusChange This bit is set whenever a connect or disconnect event occurs. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. If CurrentConnectStatus is cleared when a SetPortReset, SetPortEnable, or SetPortSuspend write occurs, this bit is set to force the driver to re-evaluate the connection status because these writes should not occur if the port is disconnected. 0 = no change in CurrentConnectStatus. 1 = change in CurrentConnectStatus. (write) ClearPortPower The HCD clears the PortPowerStatus bit by writing a 1 to this bit. Writing a 0 has no effect. **Remark:** If the DeviceRemovable[NDP] bit is set, this bit is set only after a root hub reset to inform the system that the device is attached.	0
17	PESC	PortEnableStatusChange This bit is set when hardware events cause the PortEnableStatus bit to be cleared. Changes from HCD writes do not set this bit. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. 0 = no change in PortEnableStatus. 1 = change in PortEnableStatus.	0
18	PSSC	PortSuspendStatusChange This bit is set when the full resume sequence is completed. This sequence includes the 20 ms K-state resume pulse. LS EOP, and 3 ms resychronization delay. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. This bit is also cleared when ResetStatusChange is set. 0 = resume is not completed. 1 = resume completed.	0

Table 803. Host controller root hub port status register (HCRHPORTSTATUS[1:NDP] register offset 0x54) *…continued*

Bit	Symbol	Description	Reset value
19	OCIC	PortOverCurrentIndicatorChange This bit is valid only if overcurrent conditions are reported on a per-port basis. This bit is set when root hub changes the PortOverCurrentIndicator bit. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. 0 = no change in PortOverCurrentIndicator. 1 = PortOverCurrentIndicator has changed.	0
20	PRSC	PortResetStatusChange This bit is set at the end of the 10 ms port reset signal. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. 0 = port reset is not complete. 1 = port reset is complete.	0
31:21	-	Reserved	-

42.7.23 PortMode register

The port mode register controls the host or device role in addition to setting the polarity of the ID pin.

Table 804. Port Mode (PORTMODE, offset, 0x5C)

Bit	Symbol	Description	Reset value
0	ID	Port ID pin value. This bit indicates the value on the ID line of the port This field only contains a valid value some time after the ID_EN bit is set to 1.	0
7:1	-	Reserved	-
8	ID_EN	Port ID pin pull-up enable. If this bit is set, the pull-up on the ID line of the port will be enabled. This bit must be set to read a good value in the ID field.	0
15:9	-	Reserved	-
16	DEV_ENABLE	1: Device 0: Host	0
31:17	-	Reserved	-

42.8 USB host register definitions

See the OHCI specification for more details on the OHCI registers.

43.1 How to read this chapter

The USB1 high-speed controller is available on selected LPC55S6x/LPC55S2x/LPC552x devices.

The USB1 contains the USB RAM, the only memory which the USB1 has write access to, and which enables shared access of the endpoint buffer and control data between the controller and the AHB bus. It is also possible to use this RAM as generic memory when the USB1 is not in use.

This chapter describes the host functionality of the controller.

43.2 Introduction

The USB1 high-speed controller provides a plug-and-play connection of peripheral devices to a host with three different data speeds: high-speed with a data rate of 480 Mbps, full-speed with a data rate of 12 Mbps, and low-speed with a data rate of 1.5 Mbps. Many portable devices can benefit from the ability to communicate to each other over the USB interface without intervention of a host PC.

43.2.1 Features

- Contains on-chip high-speed UTMI+ compliant transceiver (PHY).
- Supports all high-speed, full-speed, and low-speed USB-compliant peripherals.
- Complies with *Universal Serial Bus specification 2.0.*
- Supports a hardware/software interface similar to the *Enhanced Host Controller Interface* (EHCI) specification.
- Supports port power switching.
- Supports power management.
- Integrated DMA engine.

43.2.2 Architecture

Figure 158 shows the architecture of the USB host controller.

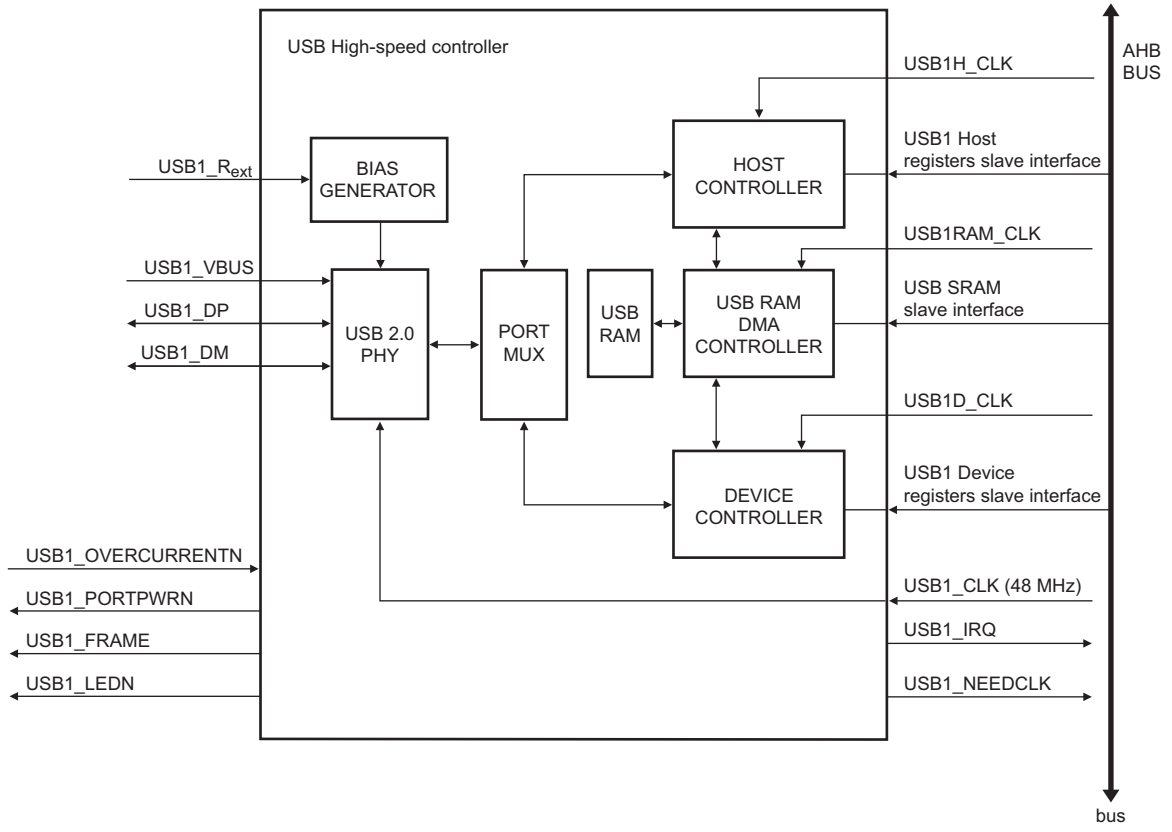


Fig 158. USB host controller block diagram

43.3 Basic configuration

Initial configuration of the USB1 host controller:

USB HS PHY: Power on and initialize the USB HS PHY. See Section 45.3 “Basic configuration”.

Pins: Configure the USB1 pins in the IOCON register block. See Table 805 and Table 340.

Clocks:

- Configure the CPU clock to a minimum frequency of 60 MHz.
- In the AHBCLKCTRL2 register, set the USB1_HOST and USB1_RAM bits.

Port Control configuration:

Set DEV_ENABLE bit to 0 in port mode register (offset 0x50) to ensure that the port is routed to USB1 host controller. See Section 43.5.19 “Port mode”.

Reset:

The USB1 host and USB RAM controller can be reset by toggling the USB1_HOST_RST and USB1_RAM_RST bits in PRESETCTRL2. See Section 4.5.9 “Peripheral reset control 2”.

Interrupts:

The USB1 has two interrupt slot assignments, one for the main interrupt, USB1_IRQ (USB1), and the other for USB1_NEEDCLK. See Table 8. Clear pending interrupts before enabling them.

Remark: Software must ensure that there is a maximum of one outstanding PTD in the list for the same device address, endpoint number, endpoint direction combination. If this rule is violated, there is a risk that the USB HS host hardware will send the packets in the wrong order.

43.4 Interfaces

43.4.1 Pin description

Table 805 describes the USB host pins.

Table 805. USB host pin description

Pin name	Port pin	IOCON function/Mode	Direction	Description
USB1_PORTPWRN	PIO1_2/ PIO1_29	7/4, pull-up	O	VBUS drive signal (towards external charge pump or power management unit); indicates that VBUS must be driven (active LOW). (Depending on the location of the pin you use, IOCON function may vary.)
USB1_OVERCURRENTN	PIO1_1/ PIO1_30	7/4, pull-up	I	Port power fault signal indicating over-current condition; this signal monitors over-current on the USB bus (external circuitry required to detect over-current condition; depending on the location of the pin you use, IOCON function may vary).
USB1_DP	-	-	I/O	Positive differential data.
USB1_DM	-	-	I/O	Negative differential data.
USB1_VDD3V3	-	-	-	USB1 analog 3.3 V supply.
USB1_VBUS	-	-	-	VBUS status input.

43.4.2 Software interface

The AHB slave interface of the USB host must be used to access the registers and to configure the mode of the port (host mode or device mode). Figure 159 shows which part of the data is stored in registers and the location of the data in the USB RAM.

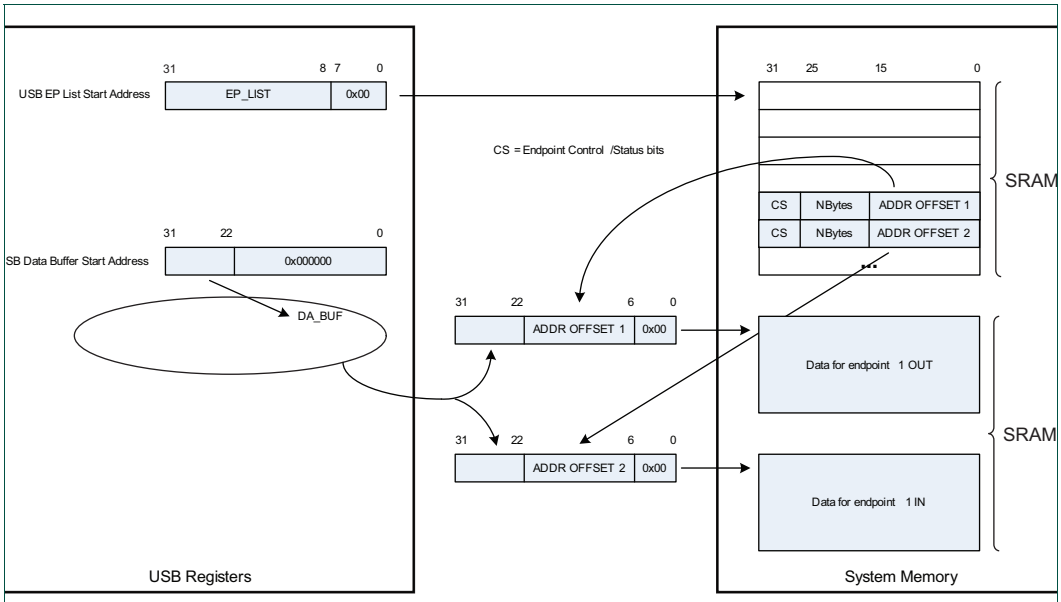


Fig 159. USB host software interface

43.5 Register description

The following registers are located in the AHB clock domain. They can be accessed directly by the processor. All registers are 32 bits wide and aligned in the word address boundaries.

Table 806. Register overview: USB high-speed device controller (base address = 0x400A 3000)

Name	Access	Offset	Description	Reset value [1]	Section
CAPLENGTH_CHIPID	RO	0x00	This register contains the offset value towards the start of the operational register space and the version number of the IP block.	0x01010010	43.5.1
HCSPARAMS	RO	0x04	Host controller structural parameters.	0x00010011	43.5.2
FLADJ_FRINDEX	R/W	0x0C	Frame length adjustment.	0x00000020	43.5.3
ATLPTD	R/W	0x10	Memory base address where ATL PTD0 is stored.	0x00000000	43.5.4
ISOPTD	R/W	0x14	Memory base address where ISO PTD0 is stored.	0x00000000	43.5.5
INTPTD	R/W	0x18	Memory base address where INT PTD0 is stored.	0x00000000	43.5.6
DATAPAYLOAD	R/W	0x1C	Memory base address that indicates the start of the data payload buffers.	0x00000000	43.5.7
USBCMD	R/W	0x20	USB command register.	0x00000000	43.5.8
USBSTS	R/W1C	0x24	USB interrupt status register.	0x00000000	43.5.9
USBINTR	R/W	0x28	USB interrupt enable register.	0x00000000	43.5.10
PORTSC1	R/W	0x2C	Port status and control register.	0x00000000	43.5.11
ATLPTDD	R/W1C	0x30	Done map for each ATL PTD.	0x00000000	43.5.12
ATLPTDS	R/W	0x34	Skip map for each ATL PTD.	0x00000000	43.5.13
ISOPTDD	R/W1C	0x38	Done map for each ISO PTD.	0x00000000	43.5.14
ISOPTDS	R/W	0x3C	Skip map for each ISO PTD.	0x00000000	43.5.15
INTPTDD	R/W1C	0x40	Done map for each INT PTD.	0x00000000	43.5.16
INTPTDS	R/W	0x44	Skip map for each INT PTD.	0x00000000	43.5.17
LASTPTD	R/W	0x48	Marks the last PTD in the list for ISO, INT and ATL.	0x00000000	43.5.18
PORTMODE	R/W	0x50	Controls the port if it is attached to the host block or the device block.	0x00040000	43.5.19

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

43.5.1 CAPLENGTH/CHIPID register

The CAPLENGTH/CHIPID register describes the capability length and the revision of the USB IP.

Table 807. Capability Length_Chip Identification register (CAPLENGTH_CHIPID, offset = 0x00)

Bits	Symbol	Description
7:0	CAPLENGTH	Capability length: It is used as an offset. It is added to the register base to find the beginning of the operational register space.
15:8	-	Reserved
31:16	CHIPID	Chip identification: Indicates major and minor revision of the IP: • [31:24] = Major revision • [23:16] = Minor revision Major revisions used: 0x01: USB2.0 high-speed host

43.5.2 HCSPARAMS register

The HCSPARAMS register describes the USB host port configuration.

Table 808. Host controller structural parameters register (HCSPARAMS, offset = 0x04)

Bits	Symbol	Description
3:0	N_PORTS	This register specifies the number of physical downstream ports implemented on this host controller. This is fixed to 0x1 for this IP.
4	PPC	This field indicates whether the host controller implementation includes port power control. The value of this bit is controlled by the generic C_PORTPOWER_CONTROL.
15:5	-	Reserved.
16	P_INDICATOR	This bit indicates whether the ports support port indicator control. The value of this bit is controlled by the generic C_PORT_INDICATORS.
31:17	-	Reserved.

43.5.3 FLADJ register (Address offset = 0x0C)

The FLADJ register controls the SOF frame length timing and the frame index.

Table 809. Frame length adjustment (FLADJ, offset = 0x0C)

Bits	Symbol	Description
5:0	FLADJ	Frame length timing value. Each decimal value change to this register corresponds to 16 high-speed bit times. The SOF cycle time (number of SOF counter clock periods to generate a SOF micro-frame length) is equal to 59488 + value in this field. The default value is decimal 32 (20h), which gives a SOF cycle time of 60000.
15:6	-	Reserved
29:16	FRINDEX	Frame index: Bits 29 to16 in this register are used for the frame number field in the SOF packet. The value in this field is incremented by one every 125 μs (independent of the speed of the attached device). Software is only allowed to update this field when the run/stop bit is set to 0.
31:30	-	Reserved.

43.5.4 ATL PTD base address register

The ATL PTD base address register configures the start address of the ATL list.

Table 810. ATL PTD base address (ATL PTD BaseAddress, offset = 0x10)

Bits	Symbol	Description
3:0	-	Reserved.
8:4	ATL_CUR	It indicates the current PTD that is used by the hardware when it is processing the ATL list.
31:9	ATL_BASE	Base address to be used by the hardware to find the start of the ATL list.

Remark: The hardware will only use the least significant bits of this register to find the correct location in the USB RAM. For example, if the USB RAM is 4 kB, bits 11 to 0 of this register will be used to find the correct byte address.

43.5.5 ISO PTD base address register

The ISO PTD base address register configures the start address of the ISO list.

Table 811. ISO PTD base address (ISO PTD BaseAddress, offset = 0x14)

Bits	Symbol	Description
4:0	-	Reserved.
9:5	ISO_FIRST	It indicates the first PTD that is used by the hardware when it is processing the ISO list.
31:10	ISO_BASE	Base address to be used by the hardware to find the start of the ISO list.

Remark: The hardware will only use the least significant bits of this register to find the correct location in the USB RAM. For example, if the USB RAM is 4 kB, bits 11 to 0 of this register will be used to find the correct byte address.

43.5.6 INT PTD base address register

The INT PTD base address register configures the start address of the INT list.

Table 812. INT PTD base address (INT PTD BaseAddress, offset = 0x18)

Bits	Symbol	Description
4:0	-	Reserved
9:5	INT_FIRST	It indicates the first PTD that is used by the hardware when it is processing the INT list.
31:10	INT_BASE	Base address to be used by the hardware to find the start of the INT list.

Remark: The hardware will only use the least significant bits of this register to find the correct location in the USB RAM. For example, if the USB RAM is 4 kB, bits 11 to 0 of this register will be used to find the correct byte address.

43.5.7 Data payload base address register

The data payload base address register configures the start address of the data payload list.

Table 813. Data payload base address (Data Payload BaseAddress, offset = 0x1C)

Bits	Symbol	Description
15:0	-	Reserved.
31:16	DAT_BASE	Base address to be used by the hardware to find the start of the data payload section. The hardware will only use the least significant bits required for addressing the correct location in the USB RAM.

43.5.8 USBCMD register

The USB command register controls the overall execution of the USB host scheduler.

Table 814. USB command register (USBCMD, offset = 0x20)

Bits	Symbol	Description
0	RS	Run/Stop: 1b = Run. The host controller executes the schedule. 0b = Stop. If this bit is set to 1b, the USB clock will be always-on.
1	HCRESET	Host controller reset: This control bit is used by the software to reset the host controller
3:2	FLS	Frame list size: This field specifies the size of the frame list. This field is used to control when the frame list roll over interrupt bit must be set. 00b 1024 elements 01b 512 elements 10b 256 elements 11b Reserved
6:4	-	Reserved.
7	LHCR	Light host controller reset: This bit allows the driver software to reset the host controller without affecting the state of the ports.
8	ATL_EN	ATL list enabled. When this bit is set, the hardware will process the ATL list.
9	ISO_EN	ISO list enabled. When this bit is set, the hardware will process the ISO list.
10	INT_EN	INT list enabled. When this bit is set, the hardware will process the INT list.
23:11	-	Reserved.
31:24	-	Reserved

43.5.9 USBSTS register

The USB interrupt status register shows the interrupt status.

Table 815. USB interrupt status register (USBSTS, offset = 0x24)

Bits	Symbol	Description
1:0	-	Reserved.
2	PCD	Port change detect: The host controller sets this bit to logic 1 when any port has a change bit transition from a 0 to a one or a force port resume bit transition from a 0 to a 1 as a result of a J-K transition detected on a suspended port. Software must write a one to clear the bit.
3	FLR	Frame list rollover: The host controller sets this bit to logic 1 when the frame list index rolls over its maximum value to 0. Software must write a one to clear the bit.
15:4	-	Reserved.
16	ATL_IRQ	ATL IRQ: Indicates that an ATL PTD (with I-bit set) was completed. The hardware interrupt line will be asserted if the respective enable bit in the USBINTR register is set. 0 - No ATL PTD event occurred. 1 - ATL PTD event occurred. Software must write a one to clear the bit.

Table 815. USB interrupt status register (USBSTS, offset = 0x24) *…continued*

Bits	Symbol	Description
17	ISO_IRQ	ISO IRQ: Indicates that an ISO PTD (with I-bit set) was completed. The hardware interrupt line will be asserted if the respective enable bit in the USBINTR register is set. 0 - No ISO PTD event occurred. 1 - ISO PTD event occurred. Software must write a one to clear the bit.
18	INT_IRQ	INT IRQ: Indicates that an INT PTD (with I-bit set) was completed. The hardware interrupt line will be asserted if the respective enable bit in the USBINTR register is set. 0 - No INT PTD event occurred. 1 - INT PTD event occurred. Software must write a one to clear the bit.
19	SOF_IRQ	SOF interrupt: Every time when the host sends a Start of Frame token on the USB bus, this bit is set. Software must write a one to clear the bit.
31:20	-	Reserved.

43.5.10 USBINTR register

The USB interrupt enable register enables or disables the interrupt. If the enable bit is set to one and the corresponding USBSTS bit is set to one, a hardware interrupt is generated.

Table 816. USB interrupt enable register (USBINTR, offset = 0x28)

Bits	Symbol	Description
1:0	-	Reserved.
2	PCDE	Port change detect interrupt enable: 1: enable 0: disable
3	FLRE	Frame list rollover Interrupt enable: 1: enable 0: disable
15:4	-	Reserved.
16	ATL_IRQ_E	ATL IRQ enable bit: 1: enable 0: disable
17	ISO_IRQ_E	ISO IRQ enable bit: 1: enable 0: disable
18	INT_IRQ_E	INT IRQ enable bit: 1: enable 0: disable
19	SOF_E	SOF interrupt enable bit: 1: enable 0: disable
31:20	-	Reserved.

43.5.11 PORTSC1 register

The port status and control register indicates the port status and configures the port operation.

Table 817. PORTSC1 register (PORTSC1, offset = 0x2C)

Bits	Symbol	Description
0	CCS	Current connect status: Logic 1 indicates a device is present on the port. Logic 0 indicates no device is present. This field is 0 if port power is 0.
1	CSC	Connect status change: Logic 1 means that the value of CCS has changed. Logic 0 means no change. This field is 0 if port power is 0. Software must write a logic 1 to clear the bit.
2	PED	Port enabled/disabled. Logic 1 means port enabled. Logic 0 means disabled. This field is 0 if port power is 0. Firmware can clear the bit to disable the port. Firmware cannot set the bit. This bit will be set at the end of a port reset sequence.
3	PEDC	Port enabled/disabled change: Logic 1 means that the value of PED has changed. Logic 0 means no change. This field is 0 if port power is 0. Software must write a logic 1 to clear the bit.
4	OCA	Over-current active: Logic 1 means that this port has an over-current condition. This bit will automatically move from one to 0 when the over-current condition is removed.
5	OCC	Over-current change: Logic 1 means that the value of OCA has changed. Logic 0 means no change. Software must write a logic 1 to clear the bit.
6	FPR	Force port resume: Logic 1 means resume (K-state) detected or driven on the port. Logic 0 means no resume detected or driven on the port. The resume signaling is driven on the port as long as this bit remains a one. For legacy (L2) transitions, software must appropriately time the resume and set this bit to a 0 when the appropriate amount of time has elapsed. This field is 0 if port power is 0.
7	SUSP	Suspend: Logic 1 means port is in the suspend state. Logic 0 means the port is not suspended. Software writes a logic 1 to this bit to put an enabled port in the L2 suspend state. Which suspend state the host controller attempts depends on the value of the Suspend Using L1 field. When in the suspend state, downstream propagation of data is blocked on this port, except for port reset. If this bit is set to a one when a transaction is in progress then the blocking will not occur until the end of the current transaction. A write of 0 is ignored by the hardware. The hardware will unconditionally set this bit to 0 when: Software sets the force port resume bit to 0. Software sets the port reset bit to a one. This field is 0 if port power is 0 or current connect status is 0.
8	PR	Port reset: Logic 1 means the port is in the reset state. Logic 0 means the port is not in reset. Software writes a logic 1 to indicate the start of the reset. SW writes a logic 0 to end the reset sequence. If the reset sequence on the USB bus is finished HW will clear the bit. SW should only check the PSPD field to know the speed of the attached device when the port Reset bit is 0. This field is 0 if port power is 0.
9	-	Reserved. Read value is undefined, only zero should be written.
11:10	LS	Line status: This field reflects the current logical levels of the DP (bit 11) and DM (bit 10) signal lines.

Table 817. PORTSC1 register (PORTSC1, offset = 0x2C) *…continued*

Bits	Symbol	Description
12	PP	Port power: The function of this bit depends on the value of the Port Power Control (PPC) bit in the HCSPARAMS register. If PPC = 0b, this bit (PP) is read-only and will always be set to 1b. If PPC = 1b, this bit (PP) is RW, If the bit is set to 0, the port is not powered. If the bit is set to one the port is powered.
13	-	Reserved
15:14	PIC	Port indicator control : Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is logic 0. If P_INDICATOR is set to one, these bits will indicate the value of the port indicators: 00b: Port indicators are off 01b: Amber 10b: Green 11b: Undefined This field is 0 if port power is 0.
19:16	PTC	Port test control: A non-zero value indicates that the port is operating in the test mode as indicated by the value. 0000b: Test mode not enabled 0001b: Test J_STATE 0010b: Test K_STATE 0011b: TEST SE0_NAK 0100b: Test_Packet 0101b: Test Force_Enable 0110b – 1111b: Reserved The reserved values should not be written by software.
21:20	PSPD	Port speed: 00b: Low-speed 01b: Full-speed 10b: High-speed 11b: Reserved
22	WOO	Wake on overcurrent enable: Writing this bit to a one enables the port to be sensitive to overcurrent conditions as wake-up events. This field is 0 if port power is 0.
31:23	-	Reserved. A 0 value indicates that no device is present or support for the LPM feature is not present on this device.

43.5.12 ATL PTD done map register

The ATL PTD done map register represents a direct map of the done status of the 32 ATL PTDs.

Table 818. ATL PTD done map register (ATL_DONE, offset = 0x30)

Bits	Symbol	Description
31:0	ATL_DONE	The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the done map register will clear the bit.

43.5.13 ATL PTD skip map register

Table 819. ATL PTD skip map register (ATL_SKIP, offset = 0x34)

Bits	Symbol	Description
31:0	ATL_SKIP	When a bit in the PTD skip map is set to logic 1, the corresponding PTD will be skipped, independent of the V bit setting. The information in that PTD is not processed. Hardware will go automatically to the next PTD.

43.5.14 ISO PTD done map register

The ISO PTD done map register represents a direct map of the done status of the 32 ISO PTDs.

Table 820. ISO PTD done map register (ISO_DONE, offset = 0x38)

Bits	Symbol	Description
31:0	ISO_DONE	The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the done map register will clear the bit.

43.5.15 ISO PTD skip map register

The ISO PTD skip map register represents a direct map of the done status of the 32 INT PTDs.

Table 821. ISO PTD skip map register (ISO_SKIP, offset = 0x3C)

Bits	Symbol	Description
31:0	ISO_SKIP	The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the done map register will clear the bit.

43.5.16 INT PTD done map register

The INT PTD done map register represents a direct map of the done status of the 32 INT PTDs.

Table 822. INT PTD done map register (INT_DONE, offset = 0x40)

Bits	Symbol	Description
31:0	INT_DONE	The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the done map register will clear the bit.

43.5.17 INT PTD skip map register

Table 823. INT PTD skip map register (INT_SKIP, offset = 0x44)

Bits	Symbol	Description
31:0	INT_SKIP	When a bit in the PTD skip map is set to logic 1, the corresponding PTD will be skipped, independent of the V bit setting. The information in that PTD is not processed. Hardware will go automatically to the next PTD.

43.5.18 Last PTD in use register

The Last PTD in use register indicates the last PTD in the ATL list.

Table 824. Last PTD in use register (LAST PTD, offset = 0x48)

Bits	Symbol	Description
4:0	ATL_LAST	If hardware has reached this PTD and the J bit is not set, it will go to PTD0 as the next PTD to be processed.
7:5	-	Reserved
12:8	ISO_LAST	This indicates the last PTD in the ISO list. If hardware has reached this PTD, it will continue with processing the INT list
15:13	R	Reserved
20:16	INT_LAST	This indicates the last PTD in the INT list. If hardware has reached this PTD, it will continue with processing the ATL list.
31:21	-	Reserved

43.5.19 Port mode

The port mode register controls the host or device role in addition to setting the polarity of the ID pin.

Table 825. Port mode register (PortMode, offset = 0x50)

Bits	Symbol	Description
15:0	R	Reserved
16	DEV_ENABLE	If this bit is set to one, one of the ports will behave as a USB device. The DEV_ROUTE bit determines which port will be routed to the device block. The other port will be routed to the host block. If this bit is set to 0, both ports will be controlled by the USB host block.
17	-	Reserved
18	SW_CTRL_PDCOM	This bit indicates if the PHY power-down input is controlled by software or by hardware. 0b: hardware state machine controls PHY power-down. 1b: software controls PHY power-down by writing to SW_PDCOM bit.
19	SW_PDCOM	This bit is only used when SW_CTRL_PDCOM is set to 1b. When SW_CTRL_PDCOM is set to 1b, the software can directly control the PHY power-down bit by writing to this bit. 0b: PHY operational. 1b: PHY in power-down mode.
31:20	-	Reserved

43.6 USB PHY low-power operation

The USB PHY is put in low power mode if the run/stop bit is set to 0 and the USB port is in a state that allows putting the PHY in low-power mode.

A change on the status of the port will generate a wake-up event. An over current situation on the port will only generate a wake-up event if the WOO bit is set in PORTSC1 register. See Section 43.5.11 “PORTSC1 register”. The PHY can be put in low-power mode during the following states:

Port state = disconnected and linestate indicates SE0

Port state = disabled and linestate indicates J-state

Port state = suspend and linestate indicates J-state

If the PHY is in low-power mode and there is a change on the linestate bits, the PHY is brought out of low-power mode. It remains active for at least 3 μs. If the change on linestate is only a glitch and not a valid port change, the PHY is put in low-power mode again after this time.

If the WOO bit is set in PORTSC1 register, the PHY will be started when there is an over current condition and the PHY is in low-power mode. The PHY will remain active as long as the over current condition remains.

43.7 Proprietary Transfer Descriptor (PTD)

The standard Enhanced Host Controller Interface (EHCI) data structures as described in the “Enhanced Host Controller Interface Specification for Universal Serial Bus Rev. 1.0” are optimized for the host controller.

The optimized form of EHCI data structures is necessary because the controller does not have an AHB master interface. Instead, the controller exclusively uses its internal USB RAM to store and manage these data structures.

The controller manages schedules in two lists: periodic and asynchronous. The data structures are designed to provide the maximum flexibility required by USB, minimize memory traffic, and reduce hardware and software complexity. The USB host controller executes transactions for devices by using a simple shared-memory schedule. This schedule consists of data structures organized into three lists:

- qISO — Isochronous transfer.
- qINTL — Interrupt transfer.
- qATL — Asynchronous transfer; for the control and bulk transfers.

The system software maintains two lists for the host controller: periodic and asynchronous.

The high speed host controller has a maximum of 32 ISO, 32 INTL, and 32 ATL PTDs. These PTDs are used as channels to transfer data from the shared memory to the USB bus. These channels are allocated and de-allocated on receiving the transfer from the core USB driver.

Multiple transfers are scheduled to the shared memory for various endpoints by traversing the next link pointer provided by endpoint data structures, until it reaches the end of the endpoint list. There are three endpoint lists: one for ISO endpoints, and the other for INTL and ATL endpoints. If the schedule is enabled, the host controller executes the ISO schedule, followed by the INTL schedule, and then the ATL schedule.

These lists are traversed and scheduled by the software according to the EHCI traversal rule. The host controller executes the scheduled ISO, INTL and ATL PTDs. The completion of a transfer is indicated to the software by the interrupt that can be grouped under various PTDs by using the AND or OR registers that are available for each schedule type: ISO, INTL and ATL. These registers are simple logic registers to decide the completion status of group and individual PTDs. When the logical conditions of the Done bit is true in the shared memory, it means that PTD has completed.

There are four types of interrupts in the high speed host controller: ISO, INTL, ATL and SOF. The NextPTD pointer is a feature that allows the high speed host controller to jump unused and skip PTDs. This will improve the PTD transversal latency time. The NextPTD pointer is not meant for same or single endpoint. The NextPTD works only in forward direction.

The NextPTD traversal rules defined by the high speed host controller are:

1. Start the PTD memory vertical traversal, considering the skip and LastPTD information.
2. If the current PTD is active and not done, perform the transaction.
3. Follow the NextPTD pointer as specified in bits 4 to 0 of DW4.
4. If combined with LastPTD, the LastPTD setting must be at a higher address than the NextPTD specified.
5. If combined with skip, the skip must not be set (logically) on the same position corresponding to NextPTD, pointed by the NextPTD pointer.
6. If PTD is set for skip, it will be neglected and the next vertical PTD will be considered.
7. If the skipped PTD already has a setting including a NextPTD pointer that will not be taken into consideration, the behavior will be the same as described in step 6.

Figure 160 shows the flowchart of the PTD scheduler.

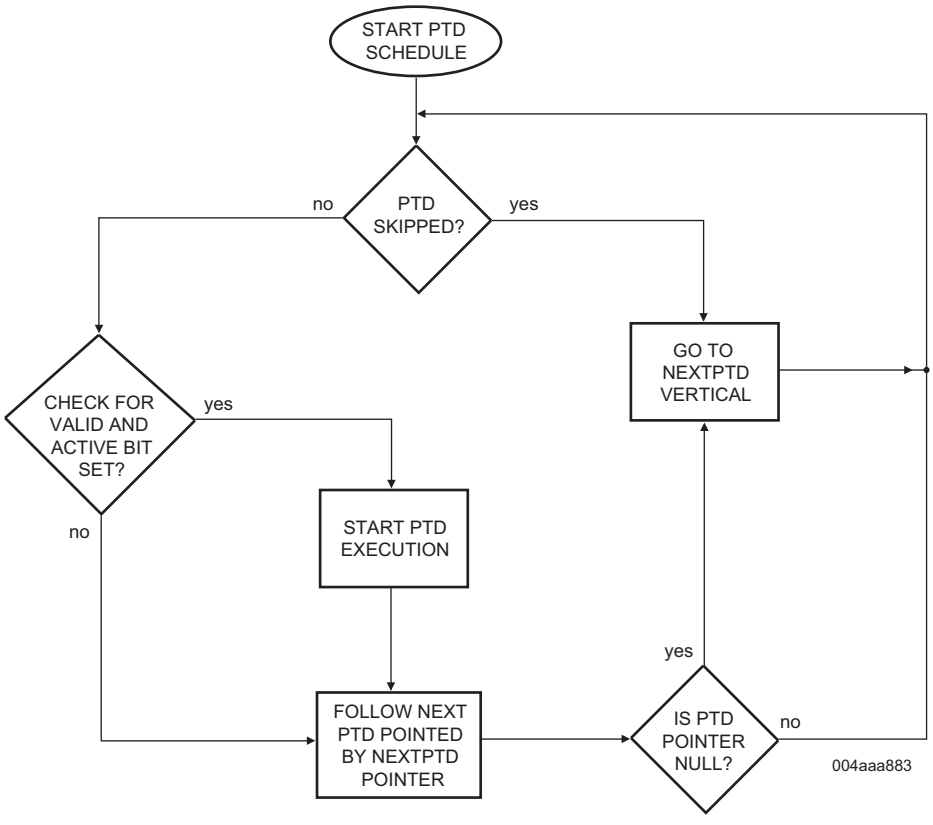


Fig 160. PTD scheduler flowchart

43.7.1 PTD on asynchronous list (qATL)

Table 826 shows the bit allocation of the bulk IN and OUT, asynchronous Transfer Descriptor (ATL PTD). This data structure is used for both regular HS/FS transactions and split transactions.

Table 826. PTD on asynchronous list (regular and split transaction)

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
R		Mult [1:0]		R	MaxPacketLength[10:0]											uFrame[7:0]								J	R	NextPTDPointer [4:0]					V	0x00
HubAddress[6:0]							PortNumber[6:0]							SE [1:0]		RL[3:0]				S	DeviceAddress[6:0]							EP[3:0]				0x04
DataStartAddress[15:0]																I	NrBytesToTransfer[14:0]															0x08
A	H	B	X	SC	P	DT	Cerr [1:0]		NakCnt[3:0				EP Type [1:0]		Token [1:0]	NrBytesToTransferred[14:0]																		0x0C

43.7.2 PTD on periodic list for regular transactions

Table 827 shows the bit allocation of the periodic IN and OUT, periodic transfer descriptor. This data structure is used for regular HS/FS transactions.

Table 827. PTD on periodic list (regular transaction)

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
R		Mult [1:0]		R	MaxPacketLength[10:0]											uFrame[7:0]								J	R	NextPTDPointer [4:0]					V	0x00
R														SE [1:0]		RL[3:0]				S	DeviceAddress[6:0]							EP[3:0]				0x04
DataStartAddress[15:0]																I	NrBytesToTransfer[14:0]															0x08
A	H	B	X	SC	P	DT	Cerr [1:0]		NakCnt[3:0				EP Type [1:0]		Token [1:0]	NrBytesToTransferred[14:0]																		0x0C
Status7 [2:0]			Status6 [2:0]			Status5 [2:0]			Status4 [2:0]			Status3 [2:0]			Status2 [2:0]		Status1 [2:0]			Status0 [2:0]			uSA[7:0]								0x10
ISO_IN_2[7:0]							ISO_IN_1[11:0]													ISO_IN_0[11:0]												0x14
ISO_IN_5 [3:0]				ISO_IN_4[11:0]												ISO_IN_3[11:0]														ISO_IN_2 [11:8]			0x18
ISO_IN_7[11:0]												ISO_IN_6[11:0]												ISO_IN_5[11:4]								0x1C

43.7.3 PTD on periodic list for split transactions

Table 828 shows the bit allocation of the periodic IN and OUT, periodic transfer descriptor. This data structure is used for split transactions on the periodic list.

Table 828. PTD on periodic list (split transaction)

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
R		Mult [1:0]		R	TT_MPS_LEN[10:0] / MaxPacketLength[10:0]											uFrame[7:0]								J	R	NextPTDPointer [4:0]					V	0x00
HubAddress[6:0]							PortNumber[6:0]							SE[1:0]		RL[3:0]				S	DeviceAddress[6:0]							EP[3:0]				0x04
DataStartAddress[15:0]																I	NrBytesToTransfer[14:0]															0x08
A	H	B	X	SC	P	DT	Cerr [1:0]		NakCnt[3:0]				EP Type [1:0]		Token [1:0]		NrBytesToTransferred[14:0]																	0x0C
Status7 [2:0]			Status6 [2:0]			Status5 [2:0]			Status4 [2:0]			Status3 [2:0]			Status2 [2:0]			Status1 [2:0]			Status0 [2:0]			uSA[7:0]								0x10
SP_ISO_IN_2[7:0]								SP_ISO_IN_1[7:0]								SP_ISO_IN_0[7:0] / S_Bytes[7:0]								uSCS[7:0]								0x14
SP_ISO_IN_6[7:0]								SP_ISO_IN_5[7:0]								SP_ISO_IN_4[7:0]								SP_ISO_IN_3[7:0]								0x18
R																								SP_ISO_IN_7[7:0]								0x1C

43.7.4 PTD bit definition

Table 829. PTD bit definition

Symbol	Access	Description
V	SW - sets HW - resets	Valid: 0b: This bit is deactivated when the entire PTD is executed (Active bit is cleared), or when a error is encountered (Halt bit is set). 1b: Software updates to one when there is payload to be sent or received. The current PTD is active.
NrBytesToTransfer[14:0]	SW - writes	Number of bytes to transfer: This field indicates the number of bytes that can be transferred by this data structure. It is used to indicate the depth of the DATA field (32 kB - 1).
MaxPacketLength[10:0] TT_MPS_Len[10:0]	SW - writes	Transaction translator maximum packet size length: This field indicates the maximum number of bytes that can be sent per start split, depending on the number of total bytes needed. If the total bytes to be sent for the entire millisecond is greater than 188 bytes, this field should be set to 188 bytes for an OUT token and 192 bytes for an IN token. Otherwise, this field should be equal to the total bytes sent.
Mult[1:0]	SW - writes	Multiplier: This field is a multiplier used by the host controller as the number of successive packets the host controller may submit to the endpoint in the current execution. Note: Only 01 value is supported. Please refer to the Chip Errata document for further details.
I	SW - writes	Interrupt on complete: If this bit is set and the PTD is completed, the interrupt status bit of the corresponding list is set to one.
EP[3:0]	SW - writes	Endpoint: This is the USB address of the endpoint within the function.
DeviceAddress[6:0]	SW - writes	Device address: This is the USB address of the function containing the endpoint that is referred to by this buffer.

Table 829. PTD bit definition *…continued*

Symbol	Access	Description
S	SW - writes	This bit indicates whether a split transaction has to be executed. 0 – No split transaction – speed is same as port speed. 1 – Split transaction.
Token[1:0]	SW - writes	Token: Identifies the token Packet Identifier (PID) for this transaction. 00 – OUT 01 – IN 10 – SETUP 11 – Undefined
EPType[1:0]	SW - writes	Transaction type: 00 – Control 01 – Isochronous 10 – Bulk 11 – Interrupt
SE[1:0]	SW - writes	This specifies the speed for a control or interrupt transaction to a device that is not high-speed: 00 – Full-speed 10 – Low-speed 01 or 11 – Undefined behavior For isochronous and bulk transactions this field must be set to 00. If a full-speed hub is connected to the port, the SE[1] bit indicates if a low-speed packet must be sent on a full-speed bus.
PortNumber[6:0]	SW - writes	Port number: This indicates the port number of the hub.
HubAddress[6:0]	SW - writes	Hub address: This indicates the hub address.
NextPTDPointer[4:0]	SW - writes	Next PTD counter: Next PTD branching assigned by the PTDpointer.
J	SW - writes	Jump: 0: increment the PTD pointer. 1: enable the next PTD branching.
DataStartAddress[15:0]	SW - writes	Data start address: “DataStartAddress + DataPayload_BaseAddress” is the start address that points to the start of the data buffer that will be sent or received on or from the USB bus. The hardware does not update this field when the transfer is completed.
uFrame[7:0]		This field is only applicable for interrupt and isochronous endpoints. Interrupt endpoint: Bits 2 to 0 of this field together with the μsA field represent the polling rate. When the polling rate is ≤ 1 ms, bits 2 to 0 are set to 0. If the polling rate is greater than 1 ms, bits 2 to 0 define the polling rate and bits 7 to 3 define the frame number when the packet must be transmitted. Isochronous endpoint: Bits 2 to 0 — Don’t care. Bits 7 to 3 — Frame number at which this PTD will be sent.
NrBytesTransferred[14:0]	HW — writes	Number of bytes transferred: This field indicates the number of bytes sent or received for this transaction. If Mult[1:0] is greater than one, it is possible to store intermediate results in this field.
RL[3:0]	SW - writes	Reload: If RL is set to 0h, hardware ignores the NakCnt value. RL and NakCnt are set to the same value before a transaction.

Table 829. PTD bit definition *…continued*

Symbol	Access	Description
NakCnt[3:0]	HW - writes SW - writes	NAK Counter: This field corresponds to the NakCnt field in TD. Software writes for the initial PTD launch. The V bit is reset if NakCnt decrements to zero and RL is a nonzero value. It reloads from RL if transaction is ACK-ed.
Cerr[1:0]	HW - writes SW - writes	Error counter: This field corresponds to the Cerr[1:0] field in TD. The default value of this field is 0 for isochronous transactions. 00 — The transaction will not retry. 11 — The transaction will retry three times. The hardware will decrement these values.
DT	HW - updates SW - writes	Data toggle: This bit is filled by software to start a PTD. If NrBytesToTransfer[14:0] is not complete, software needs to read this value and use this value to program the next PTD that will send data to the same endpoint.
P	SW - writes HW - updates	Ping: For high-speed transactions, this bit corresponds to the ping state bit in the status field of a TD. 0 — Ping is not set. 1 — Ping is set. For the first time, software sets the ping bit to 0. For the successive asynchronous TD, software sets the bit in asynchronous TD based on the state of the bit for the previous asynchronous TD of the same transfer: The current asynchronous TD is completed with the ping bit set. The next asynchronous TD will have its ping bit set by the software.
SC	SW - writes 0 HW - updates	Start/Complete: 0 — Start split 1 — Complete split
X	HW - writes	Error: This bit corresponds to the transaction error bit in the status field of iTD, siTD or TD. 0 — No PID error. 1 — If there are PID errors, this bit is set active. The A and V bits are also set to inactive. This transaction is retried three times.
B	HW - writes	Babble: This bit corresponds to the babble detected bit in the status field of iTD, siTD or TD. 1 — When babbling is detected, A and V are set to 0.
H	HW - writes	Halt: Set to a 1 by the host controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this data structure. This can be caused by babble, the error counter counting down to 0, or reception of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set to a one, the Active bit is also set to 0.
A	SW - sets	Active: This bit is the same as the valid bit.
uSA[7:0]	SW - writes (0 →1) HW - writes (1 → 0) after processing	This field is only used for periodic split transactions or if the port is enabled in HS mode. uSOF Active: When the frame number of bits uFrame[7:3] match the frame number of the USB bus, these bits are checked for 1 before they are sent for uSOF. For example: If uSA[7:0] = 1111 1111b: send ISO every uSOF of the entire millisecond. If uSA[7:0] = 0101 0101b: send ISO only on uSOF0, uSOF2, uSOF4, and SOF6.

Table 829. PTD bit definition *…continued*

Symbol	Access	Description
Status0[2:0]	HW - writes	Isochronous IN or OUT status at uSOF0 Bit 2: Underrun Bit 1: Babble Bit 0: XactErr
Status1[2:0]	HW - writes	Isochronous IN or OUT status at uSOF1.
Status2[2:0]	HW - writes	Isochronous IN or OUT status at uSOF2.
Status3[2:0]	HW - writes	Isochronous IN or OUT status at uSOF3.
Status4[2:0]	HW - writes	Isochronous IN or OUT status at uSOF4.
Status5[2:0]	HW - writes	Isochronous IN or OUT status at uSOF5.
Status6[2:0]	HW - writes	Isochronous IN or OUT status at uSOF6.
Status7[2:0]	HW - writes	Isochronous IN or OUT status at uSOF7.
ISO_IN0[11:0]	HW - writes	Bytes received during uSOF0, if uSA[0] is set to 1 and frame number is correct.
ISO_IN1[11:0]	HW - writes	Bytes received during uSOF1, if uSA[1] is set to 1 and frame number is correct.
ISO_IN2[11:0]	HW - writes	Bytes received during uSOF2, if uSA[2] is set to 2 and frame number is correct.
ISO_IN3[11:0]	HW - writes	Bytes received during uSOF3, if uSA[3] is set to 3 and frame number is correct.
ISO_IN4[11:0]	HW - writes	Bytes received during uSOF4 if uSA[4] is set to 4 and frame number is correct.
ISO_IN5[11:0]	HW - writes	Bytes received during uSOF5 if uSA[5] is set to 5 and frame number is correct.
ISO_IN6[11:0]	HW - writes	Bytes received during uSOF6 if uSA[6] is set to 6and frame number is correct.
ISO_IN7[11:0]	HW - writes	Bytes received during uSOF7 if uSA[7] is set to 6and frame number is correct.
uSCS[7:0]	SW - writes (0 →1) HW - writes (1 → 0) after processing	All bits can be set to one for every transfer. It specifies which uSOF the complete split needs to be sent. Start split and complete split Active bits, uSA = 0000 0001b, uSCS = 0000 0100b, will cause SS to execute in uFrame0 and CS in uFrame2.
SP_ISO_IN0[7:0]	HW - writes	Bytes received during uSOF0, if uSA[0] is set to 1 and frame number is correct.
SP_ISO_IN1[7:0]	HW - writes	Bytes received during uSOF1, if uSA[1] is set to 1 and frame number is correct.
SP_ISO_IN2[7:0]	HW - writes	Bytes received during uSOF2, if uSA[2] is set to 1 and frame number is correct.
SP_ISO_IN3[7:0]	HW - writes	Bytes received during uSOF3, if uSA[3] is set to 1 and frame number is correct.
SP_ISO_IN4[7:0]	HW - writes	Bytes received during uSOF4, if uSA[4] is set to 1 and frame number is correct.
SP_ISO_IN5[7:0]	HW - writes	Bytes received during uSOF5, if uSA[5] is set to 1 and frame number is correct.

Table 829. PTD bit definition *…continued*

Symbol	Access	Description
SP_ISO_IN6[7:0]	HW - writes	Bytes received during uSOF6, if uSA[6] is set to 1 and frame number is correct.
SP_ISO_IN7[7:0]	HW - writes	Bytes received during uSOF7, if uSA[7] is set to 1 and frame number is correct.
S_Bytes	HW - writes	This field is used by HW to store an intermediate value of number of bytes received while handling a complete-split for interrupt IN transfers.

43.7.4.1 Polling rate for periodic transactions

Table 830 indicates how the hardware knows when to send a certain interrupt packet based on the polling rate. uFrame[2:0] defines the polling rate.

If set to 0 and the transaction is high-speed, the uSA determines during which uFrames a packet can be sent. If set to 0 and the transaction is a normal full-speed or low-speed, the packet will be sent during every frame. If it is set to 0 and the transaction is a full-speed or low-speed split transaction, the uSA and uCS fields will determine when to send a split transaction.

If uFrame[2:0] is different from 0 and the transaction is high-speed, the bits in uFrame[7:3] and the bits in uSA are used to know during which uFrames a packet must be sent.

If uFrame[2:0] is different from 0 and the transaction is full-speed or low-speed, the bits in uFrame[7:3] are used to know during which uFrames a packet must be sent. If uFrame[2:0] is different from 0 and the transaction is a full-speed or low-speed split transaction, the bits in uFrame[7:3] and the uSA and uCS fields will determine when to send a split transaction.

Table 830. Polling rate for periodic transactions

b	Rate	uFrame[2:0]	uFrame[7:3]	uSA[7:0]
1	1 uSOF	000b	Don’t care	1111 1111b
2	2 uSOF	000b	Don’t care	1010 1010b or 0101 0101b
3	4 uSOF	000b	Don’t care	Any 2 bits set
4	1 mS	000b	Don’t care	Any 1 bit set
5	2	001b	Bit 0 is compared with FRINDEX[3]	Any 1 bit set
6	4	010b	Bits[1:0] are compared with FRINDEX[4:3]	Any 1 bit set
7	8	011b	Bits[2:0] are compared with FRINDEX[5:3]	Any 1 bit set
8	16	100b	Bits[3:0] are compared with FRINDEX[6:3]	Any 1 bit set
9	32	101b	Bits[4:0] are compared with FRINDEX[7:3]	Any 1 bit set

44.1 How to read this chapter

The USB1 high-speed controller is available on selected LPC55S6x/LPC55S2x/LPC552x devices.

The USB1 contains the USB RAM, the only memory which the USB1 has write access to, and which enables shared access of the endpoint buffer and control data between the controller and the AHB bus. It is also possible to use this RAM as generic memory when the USB1 is not in use.

Note: USB SRAM is mapped in the so-called “Peripheral” Memory Region of both CPU0 and CPU1. The “Peripheral” memory region (which extends from 0x4000_0000 to 0x5FFF_FFFF) is considered as “Device” type by default and USB RAM is located from 0x4010_0000 to 0x4010_3FFF. “Device” memory regions do not allow “unaligned” accesses by default. However, by using the Memory Protection Unit (MPU), the “Memory Type” of an address space (except for the last 0.5 GB of the 4G space) can be modified.

This chapter describes the device functionality of the controller.

44.2 Features

- USB2.0 high-speed device controller.
- Supports 12 physical (6 logical) endpoints including control endpoints.
- Supports single and double buffering.
- 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 Link Power Management (LPM).

44.3 Basic configuration

Initial configuration of the USB1 device controller:

- USB HS PHY:
  - Power on and initialize the USB HS PHY. See Section 45.3 “Basic configuration”.
- Pins: Configure the USB1 pins in the IOCON register block. See Table 832.
- Clocks:
  - Configure the CPU clock to a minimum frequency of 60 MHz, or higher.
  - Set the USB1_DEV and USB1_RAM bits in the AHBCLKCTRL2 register to enable USB device configuration and operation. See Section 4.5.19.
- Enable device control of the USB1 port:
  - Set USB1_HOST in AHBCLKCTRL2, to allow accesses to the port mode register. See Section 4.5.19.

  - Set DEV_ENABLE in the port mode register. See Section 43.5.19 “Port mode” for more details.
  - To save power, clear USB1_HOST in AHBCLKCTRL2.
- Reset:
  - Reset the USB1 device and RAM control by toggling the USB1_DEV_RST and USB1_RAM_RST bits in PRESETCTRL2. See Section 4.5.9 for more details.
- Interrupts:
  - The USB1 has two interrupt slot assignments, one for the main interrupt, USB1_IRQ (USB1), and the other for USB1_NEEDCLK. See Table 8 “Connection of interrupt sources to the NVIC” Clear pending interrupts before enabling them.
- Configure the USB1 wake-up signal (see Section 44.7.6) if necessary.
- Note for device revision 0A: To support Full-Speed and Low-Speed applications, it is recommended to use the USB0 Full-Speed port and the USB1 High-speed port for Device or Host. In addition, should the application require support of Low-Speed USB devices with USB High-Speed Host, then the application should insert a USB Hub between USB1 High-speed port and external USB devices. For more details on a given revision, please see the latest errata sheet.

  Remark: USB SRAM is mapped in so-called “Peripheral” Memory Region of both CPU0 & CPU1. The “Peripheral” memory region (which extends from 0x4000_0000 to 0x5FFF_FFFF) is of Type “Device” by default and USB RAM sits from 0x4010_0000 to 0x4010_3FFF. “Device” memory regions do not allow “unaligned” accesses by default. However, by using the Memory Protection Unit (MPU), one could modify the “Memory Type” of an address space (except for the last 0.5 GB of the 4G space).

44.4 General description

The Universal Serial Bus (USB) is a four-wire bus that supports communication between a host and one or more 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.

USB SRAM is mapped in the so-called “Peripheral” Memory Region of both CPU0 & CPU1. The “Peripheral” memory region (which extends from 0x4000_0000 to 0x5FFF_FFFF) is of Type “Device” by default and USB RAM is located from 0x4010_0000 to 0x4010_3FFF. By default, “Device” memory regions do not allow “unaligned” accesses. However, by using the Memory Protection Unit (MPU), the “Memory Type” of an address space (except for the last 0.5 GB of the 4G space) can be modified.

The host schedules transactions in 125 μs 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 8 logical or 16 physical endpoints including control 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.

The USB device controller enables high-speed (480 Mb/s) data exchange with a USB host controller.

Figure 161 shows the block diagram of the USB device controller.

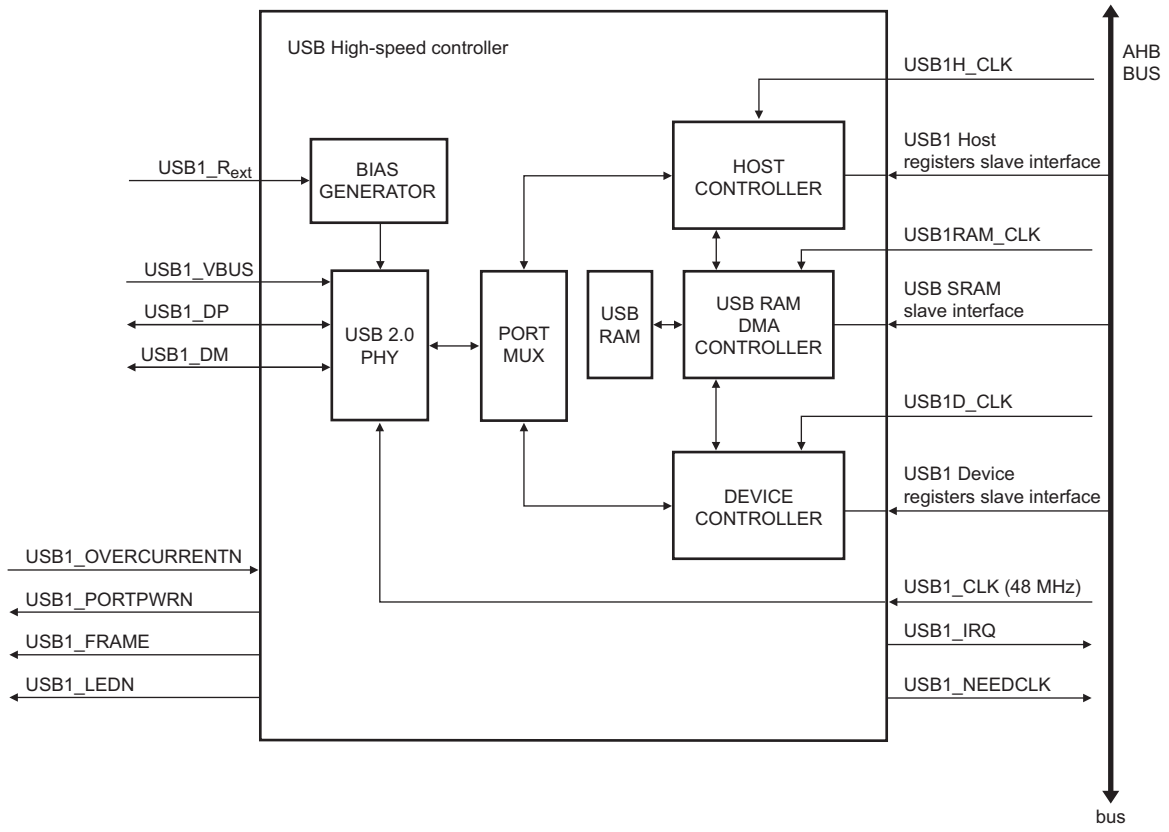


Fig 161. USB host/device controller block diagram

The USB device controller has a built-in analog transceiver (PHY). The USB PHY sends/receives the bidirectional USB1_DP and USB1_DM signals of the USB1 bus.

The Parallel Interface Engine portion of the controller implements the high-speed USB protocol layer. It is completely hard-wired 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.

44.4.1 USB1 software interface

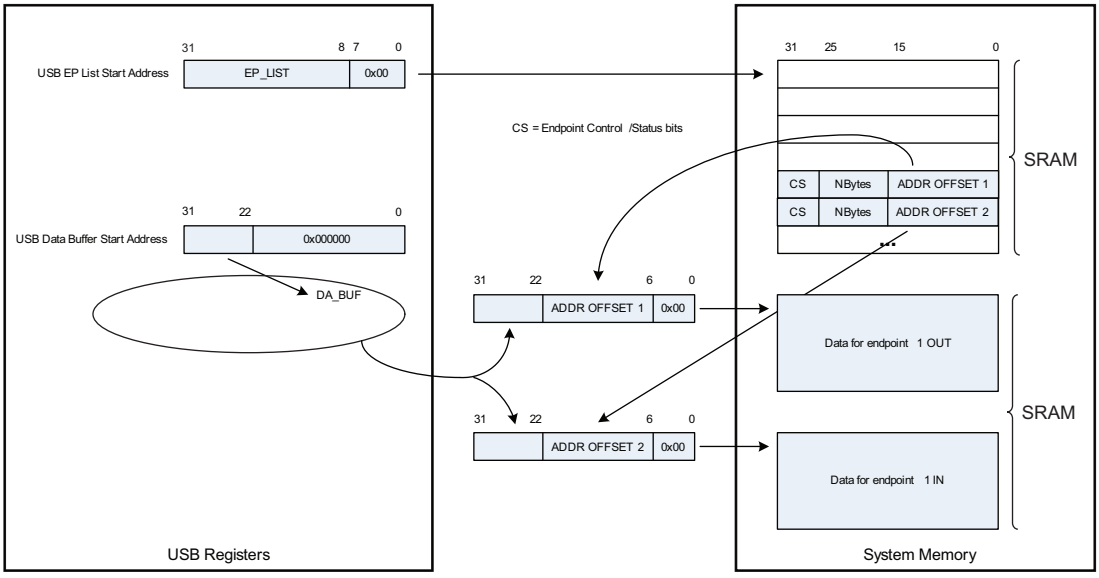


Fig 162. USB1 software interface

44.4.2 Fixed endpoint configuration

Table 831 shows the supported endpoint configurations. The packet size is configurable up to the maximum value shown in Table 831 for each type of end point.

Table 831. 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	1024/512/1024	Yes
1	3	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
2	4	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
2	5	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
3	6	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
3	7	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
4	8	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
4	9	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
5	10	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
5	11	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes

44.4.3 Interrupts

The USB controller has two interrupt lines, a general USB interrupt (USB1_IRQ [USB1]) and a USB activity wake-up interrupt (USB1_NEEDCLK). See Table 8 “Connection of interrupt sources to the NVIC”. A general 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). See Section 44.6.8 “USB1 interrupt status register” and Section 44.6.9 “USB1 interrupt enable register”.

44.4.4 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).

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 USB1 DEV_NEEDCLK signal will go low. This indicates that the USB main clock can be switched off.

When activity is detected on the USB bus, the USB1 DEV_NEEDCLK signal is activated. This process is fully combinatorial, therefore, the USB1_ DEV clock is not required to activate the USB1 DEV_NEEDCLK signal.

44.4.5 Frame toggle output

The USB1_FRAME output pin reflects the 1 kHz clock (full-speed mode) or the 8 kHz clock (high-speed mode) derived from the incoming Start of Frame tokens sent by the USB host.

44.4.6 Clocking

The USB1 device controller has the following clock connection:

- AHB clock: The AHB system bus clock controls the USB device registers, and the USB RAM DMA controller.

All other clocking is handled by the USB HS PHY. Section 45.3 “Basic configuration”.

44.5 Pin description

Table 832. USB1 device pin description

Name	Port pin	IOCON function/Mode	Direction	Description
USB1_VBUS	-	-	I	VBUS status input.
USB1_DP	-	-	I/O	Positive differential data.
USB1_DM	-	-	I/O	Negative differential data.
USB1_FRAME	PIO1_29	PIO1_29, function 5. Mode: inactive	O	USB1 frame toggle signal.
USB1_LEDN	PIO1_30	PIO1_30, function 5. Mode: inactive	O	USB1-configured LED indicator (active low).
USB1_VDD3V3	-	-	-	USB1 analog 3.3 V supply.

44.6 Register description

Table 833. Register overview: USB1 (base address = 0x4009 4000)

Name	Access	Offset	Description	Reset value	Section
DEVCMDSTAT	R/W	0x000	USB device command/status register.	0x800	44.6.1
INFO	RO	0x004	USB Info register.	0x200 0000	44.6.2
EPLISTSTART	R/W	0x008	USB EP command/status list start address.	0	44.6.3
LPM	R/W	0x010	USB link power management register.	0	44.6.4
EPSKIP	R/W	0x014	USB endpoint skip.	0	44.6.5
EPINUSE	R/W	0x018	USB endpoint buffer in use.	0	44.6.6
EPBUFCFG	R/W	0x01C	USB endpoint buffer configuration register.	0	44.6.7
INTSTAT	R/W	0x020	USB interrupt status register.	0	44.6.8
INTEN	R/W	0x024	USB interrupt enable register.	0	44.6.9
INTSETSTAT	R/W	0x028	USB set interrupt status register.	0	44.6.10
EPTOGGLE	RO	0x034	USB endpoint toggle register.	0	44.6.11

44.6.1 USB1 device command/status register

Table 834. USB1 device command/status register (DEVCMDSTAT, offset = 0x000)

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	R/W
7	DEV_EN		USB device enable. If this bit is set, the HW will start responding on packets for function address DEV_ADDR.	0	R/W
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 0, HW will handle the tokens to the CTRL EP0 as indicated by the CTRL EP0 IN and OUT data information programmed by SW.	0	R/W1C
9	FORCE_ NEEDCLK		Forces the NEEDCLK output to always be on:	0	R/W
		0	USB_NEEDCLK has normal function.		
		1	USB_NEEDCLK always 1. Clock will not be stopped in case of suspend.		
10	FORCE_VBUS	0	If this bit is set to 1, the VBUS voltage indicators from the PHY are overruled. When this bit is set, the controller will consider the VBUS to be high and signal a connect when indicated by the other bits. When this bit is low, the real V_{BUS} indications are taken into account by the controller.	0	R/W
11	LPM_SUP		LPM supported:	1	R/W
		0	LPM not supported.		
		1	LPM supported.		

Table 834. USB1 device command/status register (DEVCMDSTAT, offset = 0x000) *…continued*

Bit	Symbol	Value	Description	Reset value	Access
12	INTONNAK_AO		Interrupt on NAK for interrupt and bulk OUT EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
13	INTONNAK_AI		Interrupt on NAK for interrupt and bulk IN EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
14	INTONNAK_CO		Interrupt on NAK for control OUT EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
15	INTONNAK_CI		Interrupt on NAK for control IN EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
16	DCON		Device status - 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.	0	R/W
17	DSUS		Device status - suspend. The suspend bit indicates the current suspend state. It is set to 1 when the device has not seen any activity on its upstream port for more than 3 ms. 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	R/W
18	-		Reserved.	0	RO
19	LPM_SUS		Device status - LPM suspend. This bit represents the current LPM suspend state. It is set to 1 by hardware 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 0 to this bit, the device will generate a remote walk-up. Software can only write a 0 to this bit when the LPM_REWP bit is set to 1. Hardware resets this bit when it receives a host initiated resume. Hardware only updates the LPM_SUS bit when the LPM_SUPP bit is equal to 1.	0	R/W
20	LPM_REWP	-	LPM remote wake-up enabled by USB host. Hardware sets this bit to one when the bRemoteWake bit in the LPM extended token is set to 1. Hardware 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
21	Forced FS	0 1	0: Default - drive K chirp during reset, go through HS negotiation. 1: Force the device to FS, do not drive K chirp during reset.	0	R/W

Table 834. USB1 device command/status register (DEVCMDSTAT, offset = 0x000) *…continued*

Bit	Symbol	Value	Description	Reset value	Access
23:22	Speed		This field indicates the speed at which the device operates: 00b: reserved 01b: full-speed 10b: high-speed 11b: super-speed (reserved for future use)	01b	RO
24	DCON_C		Device status - connect change. The connect change bit is set when the pull-up resistor of the device is disconnected because VBUS disappeared. The bit is reset by writing a 1 to it.	0	R/W1C
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	R/W1C
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 1 to it.	0	R/W1C
27	-		Reserved	0	RO
28	VBUS DEBOUNCED		This bit indicates if VBUS is detected or not. The bit raises immediately when VBUS becomes high. It drops to 0 if VBUS is low for at least 3 ms. If this bit is high and the DCon bit is set, the hardware will enable the pull-up resistor to signal a connect.	0	RO
31:29	PHY_TEST_MODE		This field is written by firmware to put the PHY into a test mode as defined by the USB2.0 specification: 000b: Test mode disabled 001b: Test_J 010b: Test_K 011b: Test_SE0_NAK 100b: Test_Packet 101b: Test_Force_Enable 110b - 111b: reserved	0	R/W

44.6.2 USB1 info register

Table 835. USB1 Info register (INFO, offset = 0x004)

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	R/W
		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
23:16	Minrev	-	Minor revision	0x00	RO
31:24	Majrev	-	Major revision	0x02	RO

44.6.3 USB1 EP command/status list start address

This 32-bit register indicates the start address of the USB EP command/status list. Because the USB RAM is the only memory the device controller has the ability to write to, the USB EP command/status list must reside within this RAM.

Only a subset of these bits is programmable by software. The 8 least-significant bits are hard coded to 0 because the list must start on a 256-byte boundary. Bits 19 to 8 can be programmed by software. Bits 31:20 are hard coded to 0x401, the address of the USB RAM.

Table 836. USB1 EP command/status list start address (EPLISTSTART, offset = 0x008)

Bit	Symbol	Description	Reset value	Access
7:0	-	Reserved.	0	RO
19:8	EP_LIST_PRG	Programmable portion of the USB EP command/status list address.	0	R/W
31:20	-	Reserved.	0	RO

44.6.4 USB1 link power management register

Table 837. Link power management register (LPM, offset = 0x010)

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 0, 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

44.6.5 USB1 endpoint skip

Table 838. USB1 endpoint skip (EPSKIP, offset = 0x014)

Bit	Symbol	Description	Reset value	Access
11:0	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
31:12	-	Reserved.	0	R

44.6.6 USB1 endpoint buffer in use

Table 839. USB1 endpoint buffer in use (EPINUSE, offset = 0x018)

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved. Fixed to 0 because the control endpoint 0 is fixed to single buffering for each physical endpoint.	0	R
11:2	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
31:12	-	Reserved.	0	R

44.6.7 USB1 endpoint buffer configuration

Table 840. USB1 endpoint buffer configuration (EPBUFCFG, offset = 0x01C)

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved. Fixed to 0 because the control endpoint 0 is fixed to single buffering for each physical endpoint.	0	R
11:2	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
31:12	-	Reserved.	0	R

44.6.8 USB1 interrupt status register

Table 841. USB1 interrupt status register (INTSTAT, offset = 0x020)

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 0 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/W
1	EP0IN	Interrupt status register bit for the control EP0 IN direction. This bit will be set if NBytes transitions to 0 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/W
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 0 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/W
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 0 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/W
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 0 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/W
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 0 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/W

Table 841. USB1 interrupt status register (INTSTAT, offset = 0x020) *...continued*

Bit	Symbol	Description	Reset value	Access
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 0 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/W
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 0 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/W
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 0 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/W
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 0 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/W
10	EP5OUT	Interrupt status register bit for the EP5 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 0 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 EP5 OUT direction. Software can clear this bit by writing a one to it.	0	R/W
11	EP5IN	Interrupt status register bit for the EP5 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 0 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 EP5 IN direction. Software can clear this bit by writing a one to it.	0	R/W
29:12	-	Reserved.	-	-
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/W
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/W

44.6.9 USB1 interrupt enable register

Table 842. USB1 interrupt enable register (INTEN, offset = 0x024)

Bit	Symbol	Description	Reset value	Access
11:0	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.	0	R/W
29:12	-	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.	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.	0	R/W

44.6.10 USB1 set interrupt status register

Table 843. USB1 set interrupt status register (INTSETSTAT, offset = 0x028)

Bit	Symbol	Description	Reset value	Access
11:0	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
29:12	-	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

44.6.11 USB1 endpoint toggle

Table 844. USB1 endpoint toggle (EPTOGGLE, offset = 0x034)

Bit	Symbol	Description	Reset value	Access
29:0	TOGGLE	Endpoint data toggle: This field indicates the current value of the data toggle for the corresponding endpoint.	0	R
31:30	-	Reserved	0	R

44.7 Functional description

44.7.1 Endpoint command/status list

Figure 163 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 figure.

USB EP Command/Status FIFO start

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 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 Address Offset												0x28
A	D	S	TR	RF TV	T	EP2 IN Buffer 1 NBytes															EP2 IN Buffer Address Offset												0x2C
...																																
A	D	S	TR	RF TV	T	EP5 OUT Buffer 0 NBytes															EP5 OUT Buffer 0 Address Offset												0x50
A	D	S	TR	RF TV	T	EP5 OUT Buffer 1 NBytes															EP5 OUT Buffer 1 Address Offset												0x54
A	D	S	TR	RF TV	T	EP5 IN Buffer 0 NBytes															EP5 IN Buffer 0 Address Offset												0x58
A	D	S	TR	RF TV	T	EP5 IN Buffer 1 NBytes															EP5 IN Buffer 1 Address Offset												0x5C

aaa-021724

Fig 163. Endpoint command/status list (see also Table 845)

Table 845. Endpoint command/status bit definitions

Symbol	Access	Description
A	R/W	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 0. It will do this when it receives a short packet or when the NBytes field transitions to 0 or when software has written a one to the “skip” bit. If hardware receives a token for an endpoint that is not active, it will return the following handshake or data: Non-isochronous endpoint: NAK handshake is sent. Isochronous IN endpoint: empty data packet is sent. Isochronous OUT endpoint: received data is ignored and no handshake is sent.
D	R/W	Disabled 0: The selected endpoint is enabled. 1: The selected endpoint is disabled. When a bus reset is received, firmware must set the disable bit of all endpoints to 1. Software can only modify this bit when the active bit is 0.
S	R/W	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 0 and the stall bit is one. Software can only modify this bit when the active bit is 0.
TR	R/W	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 0, 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 0 when the endpoint is reset.
RF / TV	R/W	Rate feedback mode / Toggle value For the control endpoint 0 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. For the non-control endpoints, this bit is used together with the T-bit to identify the type of endpoint When the endpoint type (T) is set to generic endpoint, this bit selects between bulk endpoint and interrupt endpoint in rate-feedback mode. 0: Bulk endpoint with maximum packet size of 512 bytes in HS mode and 64 bytes in FS mode 1: Interrupt endpoint in ‘rate feedback mode’. This means that the data toggle is fixed to 0 for all data packets. When the interrupt endpoint is in ‘rate feedback mode’, the TR bit must always be set to 0. When the endpoint type (T) is set to periodic, this bit determines if the endpoint is interrupt or isochronous. 0: Isochronous endpoint (max packet size is determined by the smallest value when comparing NBytes field with 1024). 1: Interrupt endpoint (max packet size is determined by the smallest value when comparing NBytes field with 1024).

Table 845. Endpoint command/status bit definitions *…continued*

Symbol	Access	Description
T	R/W	Endpoint type 0: Generic endpoint. The endpoint is configured as a bulk or rate feedback interrupt endpoint. In case of an rate feedback interrupt endpoint, the Maximum Packet Size in high-speed mode can only be maximum 512 bytes. 1: Periodic endpoint. The RF / TV bit determines if the endpoint is isochronous or interrupt.
NBytes	R/W	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. **Remark:** If a short packet is received on an OUT endpoint, the active bit clears 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	R/W	Bits 16 to 6 of the buffer start address. This address offset is updated by HW after each successful reception/transmission of a packet. HW increments the original value with the rounded up integer value when the packet size is divided by 64.E.g. if a packet of 200 bytes is successfully received, the address offset will be incremented by 4. Examples: 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 also incremented by 1. If a packet with 0 bytes is received, the address offset is not incremented.

Remark: When receiving a SETUP token for endpoint 0, the HW will only read the SETUP bytes buffer address offset to know where it has to store the received SETUP bytes. HW 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 164.

44.7.2 Control endpoint 0

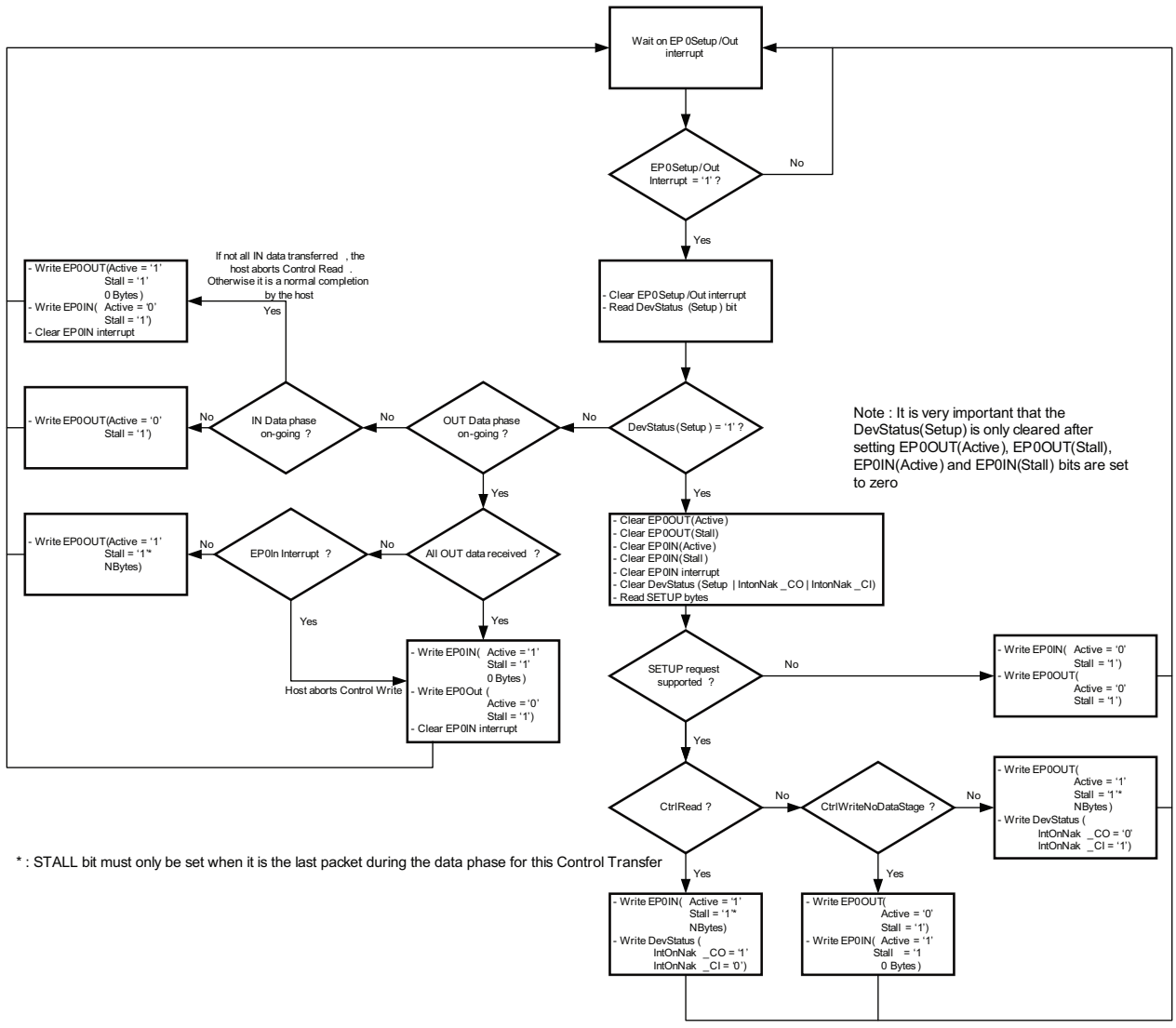


Fig 164. Flowchart of control endpoint 0 - OUT direction

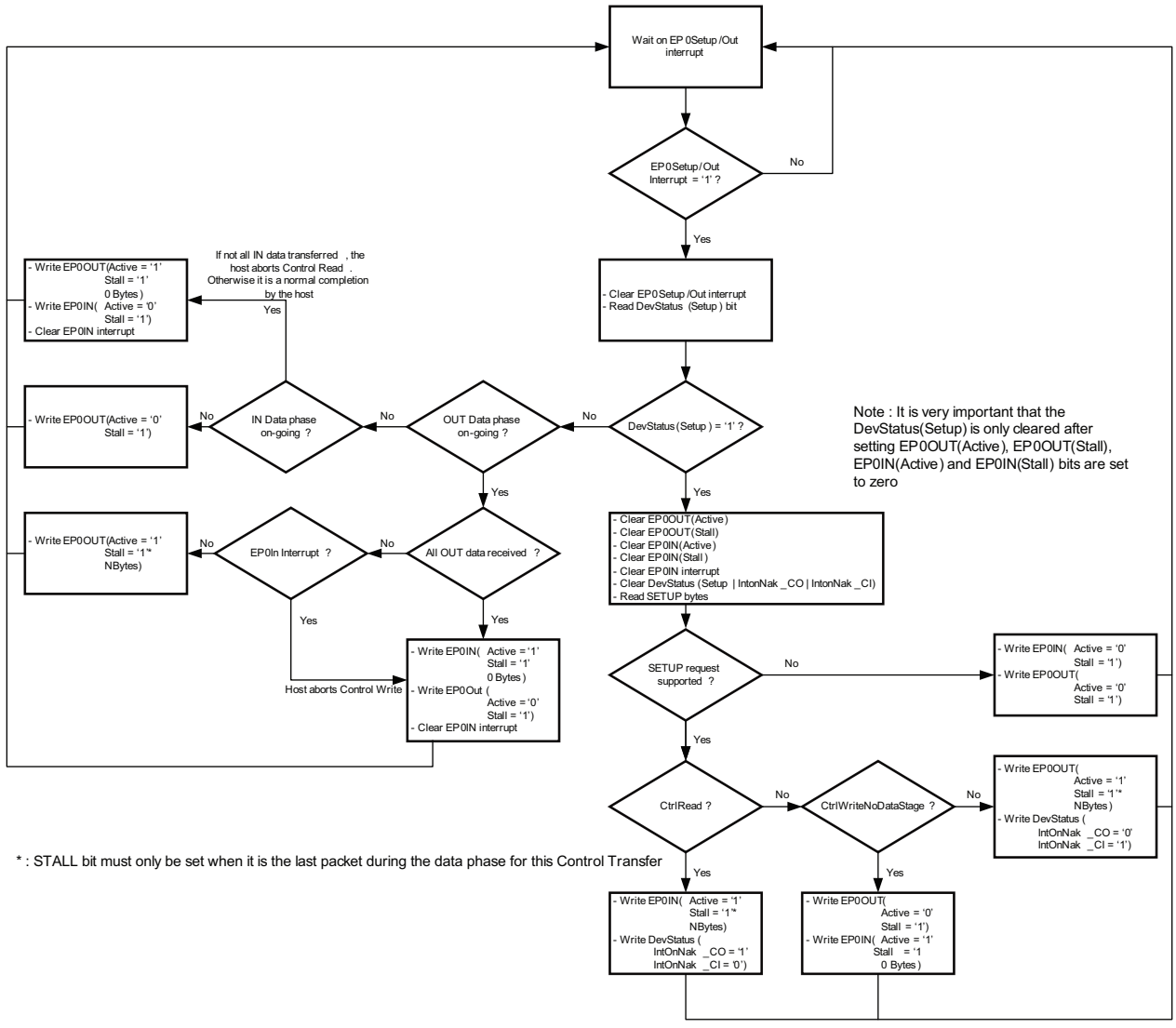


Fig 165. Flowchart of control endpoint 0 - IN direction

44.7.3 Generic endpoint: single buffering

To enable single buffering, software must set the corresponding "BUF_SB bit in the "USB EP Buffer Configuration" register" to 0. In the "USB EP Buffer in use" register, the 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 list entry for the desired endpoint and sets the active bit. The hardware will transmit/receive multiple packets for this endpoint until the NBytes value is equal to 0. When NBytes goes to 0, hardware clears the active bit and sets the corresponding endpoint interrupt status bit in INTSTAT.

Software must wait until hardware has cleared the active bit to change the command/status bits in the endpoint command/status list entry. This prevents hardware from overwriting a new value programmed by software with 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 (EPSKIP).

44.7.4 Generic endpoint: double buffering

To enable double buffering, the software must set the corresponding "USB EP Buffer Config" bit to 1. The "USB EP Buffer in use" register indicates which buffer will be used by hardware when the next token is received.

When hardware clears the active bit of the current buffer in use, it will switch the buffer in use. Software can also force hardware to use a certain buffer by writing to the corresponding "USB EP Buffer in use" bit.

44.7.5 Special cases

44.7.5.1 Use of the active bit

The use of the active bit is slightly 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.

44.7.5.2 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.

44.7.5.3 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 un-stalled and the toggle bit for that endpoint must be reset back to 0. 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.

44.7.5.4 Set configuration

When a SetConfiguration request is received with a configuration value different from 0, 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 44.7.5.3 “Clear feature (endpoint halt)” 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.

44.7.6 USB1 wake-up

44.7.6.1 Waking up from deep-sleep mode on USB activity

To allow the chip to wake-up from deep-sleep mode on USB activity, complete the following steps:

1. Set bit FORCE_NEEDCLK in the DEVCMDSTAT register, see Section 44.6.1 “USB1 device command/status register” to 0 (default) to enable automatic control of the USB1 DEV_NEEDCLK signal.
2. Set DEV_ENABLE in the PORTMODE register, port mode register and then poll USB1NEEDCLKSTAT, see Section 4.5.67 “USB1 need clock status register” until HOST_ NEEDCLK goes low.
3. Poll the DSUS bit in the DEVCMDSTAT USB1 DEV_NEEDCLK register (DSUS = 1) Section 44.6.1 “USB1 device command/status register” until the USB device is suspended. The USB1 DEV_NEEDCLK signal will be de-asserted after another 2 ms. Poll the USB1NEEDCLKSTAT register until the DEV_NEEDCLK status bit is 0.See Section 4.5.67 “USB1 need clock status register”.
4. Clear any pending USB1_NEEDCLK interrupt before enabling it. Enable the USB1_NEEDCLK in the NVIC. See Table 8.
5. Set POL_HS_DEV_NEEDCLK in the USB1NEEDCLKCTRL register to 1 to trigger the USB1_NEEDCLK activity wake-up interrupt on the rising edge of the DEV_NEEDCLK signal.
6. Enable the wake-up from deep-sleep mode on this interrupt via the POWER_EnterDeepSleep() low power API..
7. Configure the PORTMODE register, see Section 43.5.19 “Port mode” to put the PHY in power down mode:
   - Set USB1_HOST in AHBCLKCTRL2, to allow accesses to the PORTMODE register. See Section 4.5.19 “AHB clock control 2”.
   - Set SW_CTRL_PDCOM to 1 to enable software control of the PHY power down.
   - Set SW_PDCOM to 1 to put the PHY in power down mode.
   - To save power, clear USB1_HOST in AHBCLKCTRL2.

8. Enter deep-sleep mode via the power API, see Section 14.4.3 “POWER_EnterDeepSleep”. When power API is called, make sure USB HS PHY, the XO32M oscillator, and the 32k clock source are ON before going to deep-sleep mode.
9. The chip automatically wakes up and resumes execution on USB activity. After wake-up, configure the PORTMODE register (Section 43.5.19 “Port mode”) to put the PHY back to operational mode:
   - Set USB1_HOST in AHBCLKCTRL2, to allow accesses to the PORTMODE register. See Section 4.5.19 “AHB clock control 2”.
   - Set SW_CTRL_PDCOM to 1 to enable software control of the PHY power down.
   - Clear SW_PDCOM to 0 to put the PHY in operational mode.
   - To save power, clear USB1_HOST in AHBCLKCTRL2.

44.7.6.2 Remote wake-up

To issue a remote wake-up when the USB activity is suspended, complete the following steps:

1. Set bit FORCE_NEEDCLK in the DEVCMDSTAT register to 0 Section 44.6.1 “USB1 device command/status register”, default to enable automatic control of the USB1 DEV_NEEDCLK signal.
2. When it is time to issue a remote wake-up, turn on the USB1_DEV clock.
3. Force the USB clock on by writing a 1 to bit FORCE_NEEDCLK, see Section 44.6.1 “USB1 device command/status register” in the DEVCMDSTAT register.
4. Write a 0 to the DSUS bit in the DEVCMDSTAT register, see Section 44.6.1 “USB1 device command/status register”.
5. Wait until the USB device leaves the suspend state by polling the DSUS bit in the DEVCMDSTAT register (DSUS =0).

45.1 How to read this chapter

The USB1 High-Speed Physical Layer (PHY) is available on LPC55S6x/LPC55S2x/LPC552x devices that include USB high-speed controllers.

This chapter describes the functionality of the USB PHY.

45.2 Features

- USB 2.0 compliance.
- Low-speed (LS), Full-Speed (FS), and High-Speed (HS) support.
- Integrated 480 MHz PLL.

45.3 Basic configuration

Initial configuration of the USB1_PHY:

- Clocks:
  - The XO32M crystal oscillator and must be powered up and configured to one of the supported USB1_PHY reference clock frequencies of 16 MHz, 19.2 MHz, 20 MHz, 24 MHz or 32 MHz. Set the ENABLE_USB_HS_CLK_OUT bit of the XO32M_CTRL register to enable the XO32M clock output to the USB1_PHY. See Section 11.5.7.
  - Enable the 32k_osc clock to provide the 32 kHz clock to the USB1_PHY.
  - Set USB1_PHY in AHBCLKCTRL2, to enable clock to the USB1_PHY's APB register interface.
- Power: Clear the following bits to power up the USB1_PHY: See the PDRONCFG0 register for more details in Section 13.4.17.
  - PDEN_USB1_PHY to power up the USB1_PHY.
  - PDEN_LDOUSBHS to power up the USB1_PHY LDO.
- Interrupt:
  - If desired, enable the USB1_PHY interrupt. See: Table 8 "Connection of interrupt sources to the NVIC". Clear pending interrupts before enabling them.
- Reset:
  - Toggle the USB1_PHY_RST bit in PRESETCTRL2 to reset the PHY's APB registers.
- Interrupt:
  - Enable the USB1_PHY interrupt. See: Table 8 "Connection of interrupt sources to the NVIC". Clear pending interrupts before enabling them.

- Initial configuration: The following pseudo code gives an example of initializing the PHY control registers:

```
USB1_PHY_CTRL_CLR    = SFTRST;

  // Set the PLL_DIV_SEL field, USB1_PHY_PLL_SIC[24:22], to DIV_VAL
  // DIV_VAL should be set based on input frequency from XO32M.
  USB1_PHY_PLL_SIC     = (USB1_PHY_PLL_SIC & ~(0x7 << 22)) | (DIV_VAL << 22);

  USB1_PHY_PLL_SIC_SET = PLL_REG_EN;
  USB1_PHY_PLL_SIC_CLR = PLL_BYPASS;
  // add code to wait more than 15 us here
  USB1_PHY_PLL_SIC_SET = PLL_POWER;
  USB1_PHY_PLL_SIC_SET = PLL_EN_USB_CLKS;

  // enable auto power down of PHY PLL during suspend
  USB1_PHY_PLL_SIC_SET = PLL_MISC2_CONTROL0;

  USB1_PHY_CTRL_CLR    = CLKGATE;
  USB1_PHY_PWD         = 0x0;
  USB1_PHY_CTRL_SET    = ENUTMILEVEL3;
  USB1_PHY_CTRL_SET    = ENUTMILEVEL2;
  USB1_PHY_CTRL_SET    = ENAUTOCLR_CLKGATE;

  // enable using 32kHz clock for sending host resume
  USB1_PHY_CTRL_SET    = AUTORESUME_EN;

  USB1_PHY_CTRL_SET    = ENAUTOCLR_PHY_PWD;
  USB1_PHY_CTRL_SET    = ENHOSTDISCONDETECT;
```

45.4 General description

The chip contains one integrated USB 2.0 PHY Macrocell capable of connecting to USB host/device systems at the USB low-speed (LS) rate of 1.5 Mbits/s, the full-speed (FS) rate of 12 Mbits/s, or the USB 2.0 high-speed (HS) rate of 480 Mbits/s.

See Figure 166 for a block diagram of the PHY. The integrated PHY provides a standard UTMI+ interface to the USB HS controller. It has an integrated 480 MHz PLL, and an APB bus interface for configuration of its control registers. The USB_DP and USB_DM pins connect directly to a USB connector.

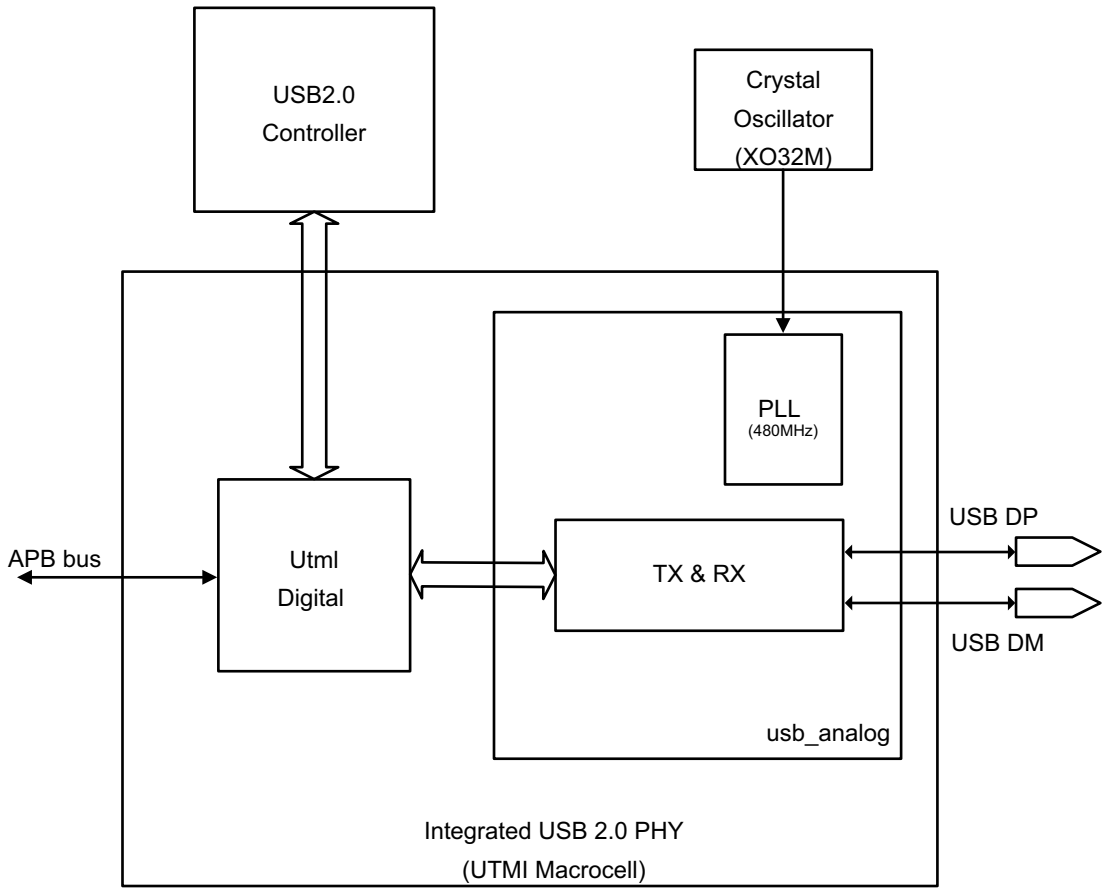


Fig 166. USB 2.0 PHY block diagram

45.5 Pin description

Table 846. USB1 High-Speed PHY pin description

Name	Port pin	IOCON function/Mode	Direction	Description
USB1_VBUS	-	-	I	VBUS status input.
USB1_DP	-	-	I/O	Positive differential data.
USB1_DM	-	-	I/O	Negative differential data.
USB1_VDD3V3	-	-	-	USB1 analog 3.3 V supply.

45.6 Register description

Table 847. Register overview: crr_d_ip_hs_usb2phy_gf40nvrf (base address = 0x50038000)

Name	Access	Offset	Description	Reset value	Section
PWD	RW	0x0	Power-down register.	0x1E1C00	45.6.1
PWD_SET	RW	0x4	Power-down register.	0x1E1C00	45.6.2
PWD_CLR	RW	0x8	Power-down register.	0x1E1C00	45.6.3
PWD_TOG	RW	0xC	Power-down register.	0x1E1C00	45.6.4
TX	RW	0x10	Transmitter register.	0xA000402	45.6.5
TX_SET	RW	0x14	Transmitter register.	0xA000402	45.6.6
TX_CLR	RW	0x18	Transmitter register.	0xA000402	45.6.7
TX_TOG	RW	0x1C	Transmitter register.	0xA000402	45.6.8
RX	RW	0x20	Receiver register.	0x0	45.6.9
RX_SET	RW	0x24	Receiver register.	0x0	45.6.10
RX_CLR	RW	0x28	Receiver register.	0x0	45.6.11
RX_TOG	RW	0x2C	Receiver register.	0x0	45.6.12
CTRL	RW	0x30	General purpose control register.	0xC0000000	45.6.13
CTRL_SET	RW	0x34	General purpose control register.	0xC0000000	45.6.14
CTRL_CLR	RW	0x38	General purpose control register.	0xC0000000	45.6.15
CTRL_TOG	RW	0x3C	General purpose control register.	0xC0000000	45.6.16
STATUS	RW	0x40	Status register.	0x0	45.6.17
PLL_SIC	RW	0xA0	PLL SIC register.	0xD12000	45.6.18
PLL_SIC_SET	RW	0xA4	PLL SIC register.	0xD12000	45.6.19
PLL_SIC_CLR	RW	0xA8	PLL SIC register.	0xD12000	45.6.20
PLL_SIC_TOG	RW	0xAC	PLL SIC register.	0xD12000	45.6.21
USB1_VBUS_DETECT	RW	0xC0	VBUS detect register.	0x700004	45.6.22
USB1_VBUS_DETECT_SET	RW	0xC4	VBUS detect register.	0x700004	45.6.23
USB1_VBUS_DETECT_CLR	RW	0xC8	VBUS detect register.	0x700004	45.6.24
USB1_VBUS_DETECT_TOG	RW	0xCC	VBUS detect register.	0x700004	45.6.25
USB1_VBUS_DET_STAT	R	0xD0	VBUS detect register.	0x0	45.6.26
ANACTRL	RW	0x100	Analog register.	0xA000402	45.6.27
ANACTRL_SET	RW	0x104	Analog register.	0xA000402	45.6.28
ANACTRL_CLR	RW	0x108	Analog register.	0xA000402	45.6.29
ANACTRL_TOG	RW	0x10C	Analog register.	0xA000402	45.6.30

45.6.1 Power down register

Table 848. Power down register (PWD, offset = 0x0)

Bit	Symbol	Access	Description	Reset value
9:0		RO	Reserved.	0x0
10	TXPWDFS	RW	Power down USB FS drivers. This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power-down the USB full-speed drivers. This turns off the current starvation sources and puts the drivers into high-impedance output.	0x1
11	TXPWDIBIAS	RW	Power down USB PHY current bias block. This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. The current bias block is shared with the battery charge circuit. Setting this bit to 1 does not power-down the circuit unless the corresponding bit in the battery charge control is also set for power-down. • 0 - Normal operation. • 1 - Power-down the USB PHY current bias block for the transmitter. This bit should be set only when the USB is in suspend mode. This effectively powers down the entire USB transmit path.	0x1
12	TXPWDV2I	RW	Power down USB PHY V-I converter and current mirror.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. The V-I converter and current mirror circuits are shared with the battery charge circuit. Setting this bit to 1 does not power-down the circuits unless the corresponding bit in the battery charge control is also set for power-down. • 0 - Normal operation. • 1 - Power-down the USB PHY transmit V-to-I converter and the current mirror.	0x1
16:13		RO	Reserved	0x0
17	RXPWDENV	RW	Power down USB HS receiver envelope detector.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power down the USB high-speed receiver envelope detector (squelch signal).	0x1
18	RXPWD1PT1	RW	Power down USB FS differential receiver.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power down the USB full-speed differential receiver.	0x1

Table 848. Power down register (PWD, offset = 0x0) *...continued*

Bit	Symbol	Access	Description	Reset value
19	RXPWDDIFF	RW	Power down USB HS differential receiver.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power down the USB high-speed differential receiver.	0x1
20	RXPWDRX	RW	Power down USB PHY receiver except the FS differential.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power-down the entire USB PHY receiver block except for the full-speed differential receiver.	0x1
31:21		RO	Reserved	0x0

45.6.2 Power down register

Table 849. Power down register (PWD_SET, offset = 0x4)

Bit	Symbol	Access	Description	Reset value
9:0		RO	Reserved.	0x0
10	TXPWDFS	RW	Power down USB FS drivers. This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power-down the USB full-speed drivers. This turns off the current starvation sources and puts the drivers into high-impedance output.	0x1
11	TXPWDIBIAS	RW	Power down USB PHY current bias block. This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. The current bias block is shared with the battery charge circuit. Setting this bit to 1 does not power-down the circuit unless the corresponding bit in the battery charge control is also set for power-down. • 0 - Normal operation. • 1 - Power-down the USB PHY current bias block for the transmitter. This bit should be set only when the USB is in suspend mode. This effectively powers down the entire USB transmit path.	0x1
12	TXPWDV2I	RW	Power down USB PHY V-I converter and current mirror.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. The V-I converter and current mirror circuits are shared with the battery charge circuit. Setting this bit to 1 does not power-down the circuits unless the corresponding bit in the battery charge control is also set for power-down. • 0 - Normal operation. • 1 - Power-down the USB PHY transmit V-to-I converter and the current mirror.	0x1
16:13		RO	Reserved	0x0

Table 849. Power down register (PWD_SET, offset = 0x4) *…continued*

Bit	Symbol	Access	Description	Reset value
17	RXPWDENV	RW	Power down USB HS receiver envelope detector.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power down the USB high-speed receiver envelope detector (squelch signal).	0x1
18	RXPWD1PT1	RW	Power down USB FS differential receiver.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power down the USB full-speed differential receiver.	0x1
19	RXPWDDIFF	RW	Power down USB HS differential receiver.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power down the USB high-speed differential receiver.	0x1
20	RXPWDRX	RW	Power down USB PHY receiver except the FS differential.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power-down the entire USB PHY receiver block except for the full-speed differential receiver.	0x1
31:21		RO	Reserved	0x0

45.6.3 Power down register

Table 850. Power down register (PWD_CLR, offset = 0x8)

Bit	Symbol	Access	Description	Reset value
9:0		RO	Reserved.	0x0
10	TXPWDFS	RW	Power down USB FS drivers. This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power-down the USB full-speed drivers. This turns off the current starvation sources and puts the drivers into high-impedance output.	0x1
11	TXPWDIBIAS	RW	Power down USB PHY current bias block. This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. The current bias block is shared with the battery charge circuit. Setting this bit to 1 does not power-down the circuit unless the corresponding bit in the battery charge control is also set for power-down. • 0 - Normal operation. • 1 - Power-down the USB PHY current bias block for the transmitter. This bit should be set only when the USB is in suspend mode. This effectively powers down the entire USB transmit path.	0x1

Table 850. Power down register (PWD_CLR, offset = 0x8) *…continued*

Bit	Symbol	Access	Description	Reset value
12	TXPWDV2I	RW	Power down USB PHY V-I converter and current mirror.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. The V-I converter and current mirror circuits are shared with the battery charge circuit. Setting this bit to 1 does not power-down the circuits unless the corresponding bit in the battery charge control is also set for power-down. • 0 - Normal operation. • 1 - Power-down the USB PHY transmit V-to-I converter and the current mirror.	0x1
16:13		RO	Reserved	0x0
17	RXPWDENV	RW	Power down USB HS receiver envelope detector.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power down the USB high-speed receiver envelope detector (squelch signal).	0x1
18	RXPWD1PT1	RW	Power down USB FS differential receiver.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power down the USB full-speed differential receiver.	0x1
19	RXPWDDIFF	RW	Power down USB HS differential receiver.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power down the USB high-speed differential receiver.	0x1
20	RXPWDRX	RW	Power down USB PHY receiver except the FS differential.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power-down the entire USB PHY receiver block except for the full-speed differential receiver.	0x1
31:21		RO	Reserved.	0x0

45.6.4 Power down register

Table 851. Power down register (PWD_TOG, offset = 0xC)

Bit	Symbol	Access	Description	Reset value
9:0		RO	Reserved.	0x0
10	TXPWDFS	RW	Power down USB FS drivers. This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power-down the USB full-speed drivers. This turns off the current starvation sources and puts the drivers into high-impedance output.	0x1
11	TXPWDIBIAS	RW	Power down USB PHY current bias block. This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. The current bias block is shared with the battery charge circuit. Setting this bit to 1 does not power-down the circuit unless the corresponding bit in the battery charge control is also set for power-down. • 0 - Normal operation. • 1 - Power-down the USB PHY current bias block for the transmitter. This bit should be set only when the USB is in suspend mode. This effectively powers down the entire USB transmit path.	0x1
12	TXPWDV2I	RW	Power down USB PHY V-I converter and current mirror.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. The V-I converter and current mirror circuits are shared with the battery charge circuit. Setting this bit to 1 does not power-down the circuits unless the corresponding bit in the battery charge control is also set for power-down. • 0 - Normal operation. • 1 - Power-down the USB PHY transmit V-to-I converter and the current mirror.	0x1
16:13	-	RO	Reserved.	0x0
17	RXPWDENV	RW	Power down USB HS receiver envelope detector.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power down the USB high-speed receiver envelope detector (squelch signal).	0x1
18	RXPWD1PT1	RW	Power down USB FS differential receiver.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power down the USB full-speed differential receiver.	0x1

Table 851. Power down register (PWD_TOG, offset = 0xC) *…continued*

Bit	Symbol	Access	Description	Reset value
19	RXPWDDIFF	RW	Power down USB HS differential receiver.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power down the USB high-speed differential receiver.	0x1
20	RXPWDRX	RW	Power down USB PHY receiver except the FS differential.: This bit will be auto cleared if there is USB wake-up event while ENAUTOCLR_PHY_PWD bit of USBPHY_CTRL is enabled. • 0 - Normal operation. • 1 - Power-down the entire USB PHY receiver block except for the full-speed differential receiver.	0x1
31:21	-	RO	Reserved.	0x0

45.6.5 USB PHY Transmitter Control Register

Table 852. USB PHY Transmitter Control Register (TX, offset 0x10)

Bit	Symbol	Access	Description	Reset value
3:0	D_CAL	RW	HS TX output current trim Decode to trim the nominal 17.78mA current source for the High Speed TX drivers on USB_DP and USB_DM. This current is directly proportional to the amplitude of the High Speed TX eye diagram. The values in this bit field will be used to trim the output current when the TRIM0[TX_D_CAL_OVERRIDE] bit is set to its default value of 1'b1. When the TRIM0[TX_D_CAL_OVERRIDE] bit is cleared to value 1'b0, the current trim values in TRIM0[USBPHY_TX_D_CAL] will be used instead. • 0000 - Maximum current, approximately 19% above nominal. • 0111 - Nominal • 1111 - Minimum current, approximately 19% below nominal.	0x1
7:4	-	-	Reserved.	0x0
11:8	TXCAL45DM	RW	DM series termination resistance trim Decode to trim the nominal 45Ω series termination resistance to the USB_DM output pin. The values in this bit field will be used to trim the termination resistance when the TRIM0[TX_CAL45DM_OVERRIDE] bit is set to its default value of 1'b1. When the TRIM0[TX_CAL45DM_OVERRIDE] bit is cleared to value 1'b0, the resistance trim values in TRIM0[USBPHY_TX_CAL45DN] will be used instead. Maximum resistance is at value 4'b0000 and minimum resistance is at value 4'b1111. Resistance is centered by design at value 4'b0111.For this USB PHY, each incremental increasing trim value decreases the target resistance by about 3.5% compared to the next lower value. Trimming this resistance will impact both the overshoot/undershoot of the Full Speed TX output and the amplitude of the High Speed TX output.	0x4
12	-	-	Reserved.	0x0

Table 852. USB PHY Transmitter Control Register (TX, offset 0x10) *…continued*

Bit	Symbol	Access	Description	Reset value
13	TXENCAL45DN	RW	Enable resistance calibration for USB DN driver Not for customer use, for lab/test use only. Forces on the USB DN FS driver/HS termination state to allow measurement of internal series termination. FS TX bias must also be switched on, most easily by setting ANACTRL[TSTI_TESTMODE] to value 1'b1.	0x0
15:14	-	-	Reserved.	0x0
19:16	TXCAL45DP	RW	DP series termination resistance trim Decode to trim the nominal 45Ω series termination resistance to the USB_DP output pin. The values in this bit field will be used to trim the termination resistance when the TRIM0[TX_CAL45DP_OVERRIDE] bit is set to its default value of 1'b1. When the TRIM0[TX_CAL45DP_OVERRIDE] bit is cleared to value 1'b0, the resistance trim values in TRIM0[USBPHY_TX_CAL45DP] will be used instead. Maximum resistance is at value 4'b0000 and minimum resistance is at value 4'b1111. Resistance is centered by design at value 4'b0111.For this USB PHY, each incremental increasing trim value decreases the target resistance by about 3.5% compared to the next lower value. Trimming this resistance will impact both the overshoot/undershoot of the Full Speed TX output and the amplitude of the High Speed TX output.	0x0
20	-	-	Reserved.	0x0
21	TXENCAL45DP	RW	Enable resistance calibration on DP Not for customer use, for lab/test use only. Forces on the USB DP FS driver/HS termination state to allow measurement of internal series termination. FS TX bias must also be switched on, most easily by setting ANACTRL[TSTI_TESTMODE] to value 1'b1.	0x0
31:22	-	-	Reserved.	0x0

45.6.6 USB PHY Transmitter Control Register Set

Table 853. USB PHY Transmitter Control Register (TX_SET, offset 0x14)

Bit	Symbol	Access	Description	Reset value
3:0	D_CAL	RW	USB PHY transmitter control register for TX_SET. • 0 - Maximum current, approximately 19% above nominal. • 7 - Nominal. • 15 - Minimum current, approximately 19% below nominal.	0x1
7:4	-	-	Reserved.	0x0
11:8	TXCAL45DM	RW		0x4
12	-	-	Reserved.	0x0
13	TXENCAL45DN	RW		0x0
15:14	-	-	Reserved.	0x0
19:16	TXCAL45DP	RW		0x0

Table 853. USB PHY Transmitter Control Register (TX_SET, offset 0x14) *...continued*

Bit	Symbol	Access	Description	Reset value
20	-	-	Reserved.	0x0
21	TXENCAL45DP	RW		0x0
31:22	-	-	Reserved.	0x0

45.6.7 USB PHY Transmitter Control Register Clear

Table 854. USB PHY Transmitter Control Register (TX_CLR, offset 0x18)

Bit	Symbol	Access	Description	Reset value
3:0	D_CAL	RW	USB PHY transmitter control register for TX_TOG. • 0 - Maximum current, approximately 19% above nominal. • 7 - Nominal. • 15 - Minimum current, approximately 19% below nominal.	0x1
7:4	-	-	Reserved.	0x0
11:8	TXCAL45DM	RW		0x4
12	-	-	Reserved.	0x0
13	TXENCAL45DN	RW		0x0
15:14	-	-	Reserved.	0x0
19:16	TXCAL45DP	RW		0x0
20	-	-	Reserved.	0x0
21	TXENCAL45DP	RW		0x0
31:22	-	-	Reserved.	0x0

45.6.8 USB PHY Transmitter Control Register Toggle

Table 855. USB PHY Transmitter Control Register (TX_TOG, offset 0x1C)

Bit	Symbol	Access	Description	Reset value
3:0	D_CAL	RW	USB PHY transmitter control register for TX_TOG. • 0 - Maximum current, approximately 19% above nominal. • 7 - Nominal. • 15 - Minimum current, approximately 19% below nominal.	0x1
7:4	-	-	Reserved.	0x0
11:8	TXCAL45DM	RW		0x4
12	-	-	Reserved.	0x0
13	TXENCAL45DN	RW		0x0
15:14	-	-	Reserved.	0x0
19:16	TXCAL45DP	RW		0x0
20	-	-	Reserved.	0x0
21	TXENCAL45DP	RW		0x0
31:22	-	-	Reserved.	0x0

45.6.9 USB PHY Receiver Control Register

Table 856. USB PHY Receiver Control Register (RX, offset 0x20)

Bit	Symbol	Access	Description	Reset value
2:0	ENVADJ	RW	ENVADJ The ENVADJ field adjusts the trip point for the envelope detector. Values shown below are nominal DC settings to indicate effect of changing these bits. AC values measured during compliance testing will be somewhat higher. • 0 - Trip-Level Voltage is 0.1000 V. • 1 - Trip-Level Voltage is 0.1125 V. • 2 - Trip-Level Voltage is 0.1250 V. • 3 - Trip-Level Voltage is 0.0875 V. • 7 to 4 - Reserved.	0x0
3	-	-	Reserved.	0x0
6:4	DISCONADJ	RW	USB PHY Receiver Control Register RX. • 0 - Trip-Level Voltage is 0.56875 V. • 1 - Trip-Level Voltage is 0.55000 V. • 2 - Trip-Level Voltage is 0.58125 V. • 3 - Trip-Level Voltage is 0.60000 V. • 4 - Reserved. • 5 - Reserved. • 6 - Reserved. • 7 - Reserved.	0x0
21:7	-	-	Reserved.	0x0
22	RXDBYPASS	RW	RXDBYPASS. • 0 - Normal operation. • 1 - Use the output of the USB_DP single-ended receiver in place of the full-speed differential receiver.	0x0
31:23	-	-	Reserved.	0x0

45.6.10 USB PHY Receiver Control Register Set

Table 857. USB PHY Receiver Control Register (RX_SET, offset 0x24)

Bit	Symbol	Access	Description	Reset value
2:0	ENVADJ	RW	ENVADJ The ENVADJ field adjusts the trip point for the envelope detector. Values shown below are nominal DC settings to indicate effect of changing these bits. AC values measured during compliance testing will be somewhat higher. • 0 - Trip-Level Voltage is 0.1000 V. • 1 - Trip-Level Voltage is 0.1125 V. • 2 - Trip-Level Voltage is 0.1250 V. • 3 - Trip-Level Voltage is 0.0875 V. • 7 to 4 - Reserved.	0x0
3	-	-	Reserved.	0x0
6:4	DISCONADJ	RW	USB PHY Receiver Control Register RX. • 0 - Trip-Level Voltage is 0.56875 V. • 1 - Trip-Level Voltage is 0.55000 V. • 2 - Trip-Level Voltage is 0.58125 V. • 3 - Trip-Level Voltage is 0.60000 V. • 4 - Reserved. • 5 - Reserved. • 6 - Reserved. • 7 - Reserved.	0x0
21:7	-	-	Reserved.	0x0
22	RXDBYPASS	RW	RXDBYPASS. • 0 - Normal operation. • 1 - Use the output of the USB_DP single-ended receiver in place of the full-speed differential receiver.	0x0
31:23	-	-	Reserved.	0x0

45.6.11 USB PHY Receiver Control Register Clear

Table 858. USB PHY Receiver Control Register (RX_CLR, offset 0x28)

Bit	Symbol	Access	Description	Reset value
2:0	ENVADJ	RW	ENVADJ The ENVADJ field adjusts the trip point for the envelope detector. Values shown below are nominal DC settings to indicate effect of changing these bits. AC values measured during compliance testing will be somewhat higher. • 0 - Trip-Level Voltage is 0.1000 V. • 1 - Trip-Level Voltage is 0.1125 V. • 2 - Trip-Level Voltage is 0.1250 V. • 3 - Trip-Level Voltage is 0.0875 V. • 7 to 4 - Reserved.	0x0
3	-	-	Reserved.	0x0
6:4	DISCONADJ	RW	USB PHY Receiver Control Register RX. • 0 - Trip-Level Voltage is 0.56875 V. • 1 - Trip-Level Voltage is 0.55000 V. • 2 - Trip-Level Voltage is 0.58125 V. • 3 - Trip-Level Voltage is 0.60000 V. • 4 - Reserved. • 5 - Reserved. • 6 - Reserved. • 7 - Reserved.	0x0
21:7	-	-	Reserved.	0x0
22	RXDBYPASS	RW	RXDBYPASS. • 0 - Normal operation. • 1 - Use the output of the USB_DP single-ended receiver in place of the full-speed differential receiver.	0x0
31:23	-	-	Reserved.	0x0

45.6.12 USB PHY Receiver Control Register Toggle

Table 859. USB PHY Receiver Control Register (RX_TOG, offset 0x2C)

Bit	Symbol	Access	Description	Reset value
2:0	ENVADJ	RW	ENVADJ The ENVADJ field adjusts the trip point for the envelope detector. Values shown below are nominal DC settings to indicate effect of changing these bits. AC values measured during compliance testing will be somewhat higher. • 0 - Trip-Level Voltage is 0.1000 V. • 1 - Trip-Level Voltage is 0.1125 V. • 2 - Trip-Level Voltage is 0.1250 V. • 3 - Trip-Level Voltage is 0.0875 V. • 7 to 4 - Reserved.	0x0
3	-	-	Reserved.	0x0
6:4	DISCONADJ	RW	USB PHY Receiver Control Register DISCONADJ. • 0 - Trip-Level Voltage is 0.56875 V. • 1 - Trip-Level Voltage is 0.55000 V. • 2 - Trip-Level Voltage is 0.58125 V. • 3 - Trip-Level Voltage is 0.60000 V. • 4 - Reserved. • 5 - Reserved. • 6 - Reserved. • 7 - Reserved.	0x0
21:7	-	-	Reserved.	0x0
22	RXDBYPASS	RW	RXDBYPASS. • 0 - Normal operation. • 1 - Use the output of the USB_DP single-ended receiver in place of the full-speed differential receiver.	0x0
31:23	-	-	Reserved.	0x0

45.6.13 General purpose control register

Table 860. General purpose control register (CTRL, offset = 0x30)

Bit	Symbol	Access	Description	Reset value
0	-	RW	Reserved.	0x0
1	ENHOSTDISCONDETECT	RW	Disconnect detect.: For host mode, enables high-speed disconnect detector. This signal allows the override of enabling the detection that is normally done in the UTMI controller. The UTMI controller enables this circuit whenever the host sends a start-of-frame packet. It shall be set after HS device is connected.	0x0
2	ENIRQHOSTDISCON	RW	Enable IRQ for Host disconnect: Enables interrupt for detection of disconnection to Device when in high-speed host mode. This should be enabled after ENDEVPLUGINDETECT is enabled.	0x0
3	HOSTDISCONDETECT_IRQ	RW	Device disconnect indication.: Indicates that the device has disconnected in High-Speed mode. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0

Table 860. General purpose control register (CTRL, offset = 0x30) *…continued*

Bit	Symbol	Access	Description	Reset value
4	ENDEVPLUGINDETECT	RW	Enables non-standard resistive plugged-in detection.: This bit field controls connection of nominal 200kO resistors to both the USB_DP and USB_DM pins as one method of detecting when a USB cable is attached in device mode. This bit field must remain at a value of 1'b0 for normal USB data communication, or when using the USBHSDCD module for battery charger detection per the USB Battery Charger Specification Revision 1.2 or any other detection mechanism for USB cable plugin. The results of this detection method are reported in USBPHY_STATUS[6]. • 0 - Disables 200kO pull-up resistors on USB_DP and USB_DM pins (Default). • 1 - Enables 200kO pull-up resistors on USB_DP and USB_DM pins (Default).	0x0
5	DEVPLUGIN_POLARITY	RW	Device plugin polarity: For device mode, if this bit is cleared to 0, then it trips the interrupt if the device is plugged in. If set to 1, then it trips the interrupt if the device is unplugged.	0x0
7:6	-	RW	Reserved.	0x0
8	RESUMEIRQSTICKY	RW	Resume IRQ: Set to 1 will make RESUME_IRQ bit a sticky bit until software clear it. Set to 0, RESUME_IRQ only set during the wake-up period.	0x0
9	ENIRQRESUMEDETECT	RW	Enable IRQ Resume detect: Enables interrupt for detection of a non-J state on the USB line. This should only be enabled after the device has entered suspend mode.	0x0
10	RESUME_IRQ	RW	Resume IRQ: Indicates that the host is sending a wake-up after suspend. This bit is also set on a reset during suspend. Use this bit to wake up from suspend for either the resume or the reset case. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0
11	-	RW	Reserved.	0x0
12	DEVPLUGIN_IRQ	RW	Device connected indicator: Indicates that the device is connected. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0
13	-	RW	Reserved.	0x0
14	ENUTMILEVEL2	RW	Enable level 2 operation: Enables UTMI+ Level 2 operation for the USB HS PHY. This should be enabled if an Embedded Host use case needs to support a LS device.	0x0
15	ENUTMILEVEL3	RW	Enable level 3 operation: Enables UTMI+ Level 3 operation for the USB HS PHY. This should be enabled if an Embedded Host use case needs to support an external FS Hub with a LS device connected.	0x0
16	ENIRQWAKEUP	RW	Enable wake-up IRQ: Enables interrupt for the wake-up events.	0x0
17	WAKEUP_IRQ	RW	Wake-up IRQ: Indicates that there is a wak-eup event. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0
18	AUTORESUME_EN	RW	Enable auto resume: Enable the auto resume feature. When set, HW will use 32 kHz clock to send Resume to respond to the device remote wake-up (for host mode only). It's useful when PLL is off and reference clock is also powered down.	0x0

Table 860. General purpose control register (CTRL, offset = 0x30) *…continued*

Bit	Symbol	Access	Description	Reset value
19	ENAUTOCLR_CLKGATE	RW	Auto clear clock gate.: Enables the feature to auto-clear the CLKGATE bit if there is wake-up event while USB is suspended. This should be enabled if needed to support auto wake-up without software interaction.	0x0
20	ENAUTOCLR_PHY_PWD	RW	Auto clear PWD register bits.: Enables the feature to auto-clear the PWD register bits in USBPHY_PWD if there is wake-up event while USB is suspended. This should be enabled if needed to support auto wake-up without software interaction.	0x0
21	ENDPDMCHG_WKUP	RW	Enable DP DM change wake-up: Not for customer use. This bit field must remain at value 1’b0. Enables the feature to wake-up USB if DP/DM is toggled when USB is suspended.	0x0
22	-	RW	Reserved.	0x0
23	ENVBUSCHG_WKUP	RW	Enable VBUS change wake-up: Enables the feature to wake-up USB if VBUS is toggled when USB is suspended.	0x0
24	-	RW	Reserved.	0x0
25	ENAUTOCLR_USBCLKGATE	RW	Enable auto-clear USB Clock gate: Enables the feature to auto-clear the USB0_CLKGATE/USB1_CLKGATE register bit in HW_DIGCTL_CTRL if there is wake-up event on USB0/USB1 while USB0/USB1 is suspended. This should be enabled if needs to support auto wake-up without S/W’s interaction.	0x0
26	ENAUTOSET_USBCLKS	RW	Enable auto-set of USB clocks: Enables the feature to auto-clear the EN_USB_CLKS register bits in HW_CLKCTRL_PLL1CTRL0/HW_CLKCTRL_PLL1CTRL1 if there is wake-up event on USB0/USB1 while USB0/USB1 is suspended. This should be enabled if needs to support auto wake-up without S/W’s interaction.	0x0
27	-	RO	Reserved.	0x0
28	HOST_FORCE_LS_SE0	RW	FS EOP low-speed timing: Forces the next FS packet that is transmitted to have a EOP with low-speed timing. This bit is used in host mode for the resume sequence. After the packet is transferred, this bit is cleared. The design can use this function to force the LS SE0 or use the USBPHY_CTRL_UTMI_SUSPENDM to trigger this event when leaving suspend. This bit is used in conjunction with USBPHY_DEBUG_HOST_RESUME_DEBUG.	0x0
29	UTMI_SUSPENDM	RO	UTMI suspend: Used by the PHY to indicate a powered-down state. If all the power-down bits in the USBPHY_PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1 when USB controller entering into Suspend mode. UTMI_SUSPENDM is negative logic, as required by the UTMI specification.	0x0
30	CLKGATE	RW	UTMI clock gate: Gate UTMI Clocks. Clear to 0 to run clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated. Note this bit can be auto-cleared if there is any wake-up event when USB is suspended while ENAUTOCLR_CLKGATE bit of USBPHY_CTRL is enabled.	0x1
31	SFTRST	RW	Software reset: Writing a 1 to this bit will soft-reset the USBPHY_PWD, USBPHY_TX, USBPHY_RX, and USBPHY_CTRL registers. Set to 0 to release the PHY from reset.	0x1

45.6.14 General purpose control register

Table 861. General purpose control register (CTRL_SET, offset = 0x34)

Bit	Symbol	Access	Description	Reset value
0	-	RW	Reserved.	0x0
1	ENHOSTDISCONDETECT	RW	Disconnect detect.: For host mode, enables high-speed disconnect detector. This signal allows the override of enabling the detection that is normally done in the UTMI controller. The UTMI controller enables this circuit whenever the host sends a start-of-frame packet. It shall be set after HS device is connected.	0x0
2	ENIRQHOSTDISCON	RW	Enable IRQ for Host disconnect: Enables interrupt for detection of disconnection to Device when in high-speed host mode. This should be enabled after ENDEVPLUGINDETECT is enabled.	0x0
3	HOSTDISCONDETECT_IRQ	RW	Device disconnect indication.: Indicates that the device has disconnected in High-Speed mode. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0
4	ENDEVPLUGINDET	RW	Enables non-standard resistive plugged-in detection.: This bit field controls connection of nominal 200kO resistors to both the USB_DP and USB_DM pins as one method of detecting when a USB cable is attached in device mode. This bit field must remain at a value of 1'b0 for normal USB data communication, or when using the USBHSDCD module for battery charger detection per the USB Battery Charger Specification Revision 1.2 or any other detection mechanism for USB cable plugin. The results of this detection method are reported in USBPHY_STATUS[6]. • 0 - Disables 200kO pull-up resistors on USB_DP and USB_DM pins (Default). • 1 - Enables 200kO pull-up resistors on USB_DP and USB_DM pins (Default).	0x0
5	DEVPLUGIN_POLARITY	RW	Device plugin polarity: For device mode, if this bit is cleared to 0, then it trips the interrupt if the device is plugged in. If set to 1, then it trips the interrupt if the device is unplugged.	0x0
7:6	-	RO	Reserved.	0x0
8	RESUMEIRQSTICKY	RW	Resume IRQ: Set to 1 will make RESUME_IRQ bit a sticky bit until software clear it. Set to 0, RESUME_IRQ only set during the wake-up period.	0x0
9	ENIRQRESUMEDETECT	RW	Enable IRQ Resume detect: Enables interrupt for detection of a non-J state on the USB line. This should only be enabled after the device has entered suspend mode.	0x0
10	RESUME_IRQ	RW	Resume IRQ: Indicates that the host is sending a wake-up after suspend. This bit is also set on a reset during suspend. Use this bit to wake up from suspend for either the resume or the reset case. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0
11	-	RW	Reserved.	0x0
12	DEVPLUGIN_IRQ	RW	Device connected indicator: Indicates that the device is connected. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0
13	-	RW	Reserved.	0x0

Table 861. General purpose control register (CTRL_SET, offset = 0x34) *…continued*

Bit	Symbol	Access	Description	Reset value
14	ENUTMILEVEL2	RW	Enable level 2 operation: Enables UTMI+ Level 2 operation for the USB HS PHY. This should be enabled if an Embedded Host use case needs to support a LS device.	0x0
15	ENUTMILEVEL3	RW	Enable level 3 operation: Enables UTMI+ Level 3 operation for the USB HS PHY. This should be enabled if an Embedded Host use case needs to support an external FS Hub with a LS device connected.	0x0
16	ENIRQWAKEUP	RW	Enable wake-up IRQ: Enables interrupt for the wake-up events.	0x0
17	WAKEUP_IRQ	RW	Wake-up IRQ: Indicates that there is a wak-eup event. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0
18	AUTORESUME_EN	RW	Enable auto resume: Enable the auto resume feature. When set, HW will use 32 kHz clock to send Resume to respond to the device remote wake-up (for host mode only). It's useful when PLL is off and reference clock is also powered down.	0x0
19	ENAUTOCLR_CLKGATE	RW	Auto clear clock gate.: Enables the feature to auto-clear the CLKGATE bit if there is wake-up event while USB is suspended. This should be enabled if needed to support auto wake-up without software interaction.	0x0
20	ENAUTOCLR_PHY_PWD	RW	Auto clear PWD register bits.: Enables the feature to auto-clear the PWD register bits in USBPHY_PWD if there is wake-up event while USB is suspended. This should be enabled if needed to support auto wake-up without software interaction.	0x0
21	ENDPDMCHG_WKUP	RW	Enable DP DM change wake-up: Not for customer use. This bit field must remain at value 1'b0. Enables the feature to wake-up USB if DP/DM is toggled when USB is suspended.	0x0
22	-	RW	Reserved.	0x0
23	ENVBUSCHG_WKUP	RW	Enable VBUS change wake-up: Enables the feature to wake-up USB if VBUS is toggled when USB is suspended.	0x0
24	-	RW	Reserved.	0x0
25	ENAUTOCLR_USBCLKGATE	RW	Enable auto-clear USB Clock gate: Enables the feature to auto-clear the USB0_CLKGATE/USB1_CLKGATE register bit in HW_DIGCTL_CTRL if there is wake-up event on USB0/USB1 while USB0/USB1 is suspended. This should be enabled if needs to support auto wake-up without S/W's interaction.	0x0
26	ENAUTOSET_USBCLKS	RW	Enable auto-set of USB clocks: Enables the feature to auto-clear the EN_USB_CLKS register bits in HW_CLKCTRL_PLL1CTRL0/HW_CLKCTRL_PLL1CTRL1 if there is wake-up event on USB0/USB1 while USB0/USB1 is suspended. This should be enabled if needs to support auto wake-up without S/W's interaction.	0x0
27	-	RO	Reserved.	0x0

Table 861. General purpose control register (CTRL_SET, offset = 0x34) *…continued*

Bit	Symbol	Access	Description	Reset value
28	HOST_FORCE_LS_SE0	RW	FS EOP low-speed timing: Forces the next FS packet that is transmitted to have a EOP with low-speed timing. This bit is used in host mode for the resume sequence. After the packet is transferred, this bit is cleared. The design can use this function to force the LS SE0 or use the USBPHY_CTRL_UTMI_SUSPENDM to trigger this event when leaving suspend. This bit is used in conjunction with USBPHY_DEBUG_HOST_RESUME_DEBUG.	0x0
29	UTMI_SUSPENDM	RO	UTMI suspend: Used by the PHY to indicate a powered-down state. If all the power-down bits in the USBPHY_PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1 when USB controller entering into Suspend mode. UTMI_SUSPENDM is negative logic, as required by the UTMI specification.	0x0
30	CLKGATE	RW	UTMI clock gate: Gate UTMI Clocks. Clear to 0 to run clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated. Note this bit can be auto-cleared if there is any wake-up event when USB is suspended while ENAUTOCLR_CLKGATE bit of USBPHY_CTRL is enabled.	0x1
31	SFTRST	RW	Software reset: Writing a 1 to this bit will soft-reset the USBPHY_PWD, USBPHY_TX, USBPHY_RX, and USBPHY_CTRL registers. Set to 0 to release the PHY from reset.	0x1

45.6.15 General purpose control register

Table 862. General purpose control register (CTRL_CLR, offset = 0x38)

Bit	Symbol	Access	Description	Reset value
0	-	RW	Reserved.	0x0
1	ENHOSTDISCONDETECT	RW	Disconnect detect.: For host mode, enables high-speed disconnect detector. This signal allows the override of enabling the detection that is normally done in the UTMI controller. The UTMI controller enables this circuit whenever the host sends a start-of-frame packet. It shall be set after HS device is connected.	0x0
2	ENIRQHOSTDISCON	RW	Enable IRQ for Host disconnect: Enables interrupt for detection of disconnection to Device when in high-speed host mode. This should be enabled after ENDEVPLUGINDETECT is enabled.	0x0
3	HOSTDISCONDETECT_IRQ	RW	Device disconnect indication.: Indicates that the device has disconnected in High-Speed mode. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0

Table 862. General purpose control register (CTRL_CLR, offset = 0x38) *…continued*

Bit	Symbol	Access	Description	Reset value
4	ENDEVPLUGINDET	RW	Enables non-standard resistive plugged-in detection.: This bit field controls connection of nominal 200kO resistors to both the USB_DP and USB_DM pins as one method of detecting when a USB cable is attached in device mode. This bit field must remain at a value of 1'b0 for normal USB data communication, or when using the USBHSDCD module for battery charger detection per the USB Battery Charger Specification Revision 1.2 or any other detection mechanism for USB cable plugin. The results of this detection method are reported in USBPHY_STATUS[6]. • 0 - Disables 200kO pull-up resistors on USB_DP and USB_DM pins (Default). • 1 - Enables 200kO pull-up resistors on USB_DP and USB_DM pins (Default).	0x0
5	DEVPLUGIN_POLARITY	RW	Device plugin polarity: For device mode, if this bit is cleared to 0, then it trips the interrupt if the device is plugged in. If set to 1, then it trips the interrupt if the device is unplugged.	0x0
7:6	-	RO	Reserved.	0x0
8	RESUMEIRQSTICKY	RW	Resume IRQ: Set to 1 will make RESUME_IRQ bit a sticky bit until software clear it. Set to 0, RESUME_IRQ only set during the wake-up period.	0x0
9	ENIRQRESUMEDETECT	RW	Enable IRQ Resume detect: Enables interrupt for detection of a non-J state on the USB line. This should only be enabled after the device has entered suspend mode.	0x0
10	RESUME_IRQ	RW	Resume IRQ: Indicates that the host is sending a wake-up after suspend. This bit is also set on a reset during suspend. Use this bit to wake up from suspend for either the resume or the reset case. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0
11	-	RW	Reserved.	0x0
12	DEVPLUGIN_IRQ	RW	Device connected indicator: Indicates that the device is connected. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0
13	-	RW	Reserved.	0x0
14	ENUTMILEVEL2	RW	Enable level 2 operation: Enables UTMI+ Level 2 operation for the USB HS PHY. This should be enabled if an Embedded Host use case needs to support a LS device.	0x0
15	ENUTMILEVEL3	RW	Enable level 3 operation: Enables UTMI+ Level 3 operation for the USB HS PHY. This should be enabled if an Embedded Host use case needs to support an external FS Hub with a LS device connected.	0x0
16	ENIRQWAKEUP	RW	Enable wake-up IRQ: Enables interrupt for the wake-up events.	0x0
17	WAKEUP_IRQ	RW	Wake-up IRQ: Indicates that there is a wak-eup event. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0
18	AUTORESUME_EN	RW	Enable auto resume: Enable the auto resume feature. When set, HW will use 32 kHz clock to send Resume to respond to the device remote wake-up (for host mode only). It's useful when PLL is off and reference clock is also powered down.	0x0

Table 862. General purpose control register (CTRL_CLR, offset = 0x38) *…continued*

Bit	Symbol	Access	Description	Reset value
19	ENAUTOCLR_CLKGATE	RW	Auto clear clock gate.: Enables the feature to auto-clear the CLKGATE bit if there is wake-up event while USB is suspended. This should be enabled if needed to support auto wake-up without software interaction.	0x0
20	ENAUTOCLR_PHY_PWD	RW	Auto clear PWD register bits.: Enables the feature to auto-clear the PWD register bits in USBPHY_PWD if there is wake-up event while USB is suspended. This should be enabled if needed to support auto wake-up without software interaction.	0x0
21	ENDPDMCHG_WKUP	RW	Enable DP DM change wake-up: Not for customer use. This bit field must remain at value 1'b0. Enables the feature to wake-up USB if DP/DM is toggled when USB is suspended.	0x0
22	-	RW	Reserved.	0x0
23	ENVBUSCHG_WKUP	RW	Enable VBUS change wake-up: Enables the feature to wake-up USB if VBUS is toggled when USB is suspended.	0x0
24	-	RW	Reserved.	0x0
25	ENAUTOCLR_USBCLKGATE	RW	Enable auto-clear USB Clock gate: Enables the feature to auto-clear the USB0_CLKGATE/USB1_CLKGATE register bit in HW_DIGCTL_CTRL if there is wake-up event on USB0/USB1 while USB0/USB1 is suspended. This should be enabled if needs to support auto wake-up without S/W's interaction.	0x0
26	ENAUTOSET_USBCLKS	RW	Enable auto-set of USB clocks: Enables the feature to auto-clear the EN_USB_CLKS register bits in HW_CLKCTRL_PLL1CTRL0/HW_CLKCTRL_PLL1CTRL1 if there is wake-up event on USB0/USB1 while USB0/USB1 is suspended. This should be enabled if needs to support auto wake-up without S/W's interaction.	0x0
27	-	RO	Reserved.	0x0
28	HOST_FORCE_LS_SE0	RW	FS EOP low-speed timing: Forces the next FS packet that is transmitted to have a EOP with low-speed timing. This bit is used in host mode for the resume sequence. After the packet is transferred, this bit is cleared. The design can use this function to force the LS SE0 or use the USBPHY_CTRL_UTMI_SUSPENDM to trigger this event when leaving suspend. This bit is used in conjunction with USBPHY_DEBUG_HOST_RESUME_DEBUG.	0x0
29	UTMI_SUSPENDM	RO	UTMI suspend: Used by the PHY to indicate a powered-down state. If all the power-down bits in the USBPHY_PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1 when USB controller entering into Suspend mode. UTMI_SUSPENDM is negative logic, as required by the UTMI specification.	0x0
30	CLKGATE	RW	UTMI clock gate: Gate UTMI Clocks. Clear to 0 to run clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated. Note this bit can be auto-cleared if there is any wake-up event when USB is suspended while ENAUTOCLR_CLKGATE bit of USBPHY_CTRL is enabled.	0x1
31	SFTRST	RW	Software reset: Writing a 1 to this bit will soft-reset the USBPHY_PWD, USBPHY_TX, USBPHY_RX, and USBPHY_CTRL registers. Set to 0 to release the PHY from reset.	0x1

45.6.16 General purpose control register

Table 863. General purpose control register (CTRL_TOG, offset = 0x3C)

Bit	Symbol	Access	Description	Reset value
0	-	RW	Reserved.	0x0
1	ENHOSTDISCONDETECT	RW	Disconnect detect.: For host mode, enables high-speed disconnect detector. This signal allows the override of enabling the detection that is normally done in the UTMI controller. The UTMI controller enables this circuit whenever the host sends a start-of-frame packet. It shall be set after HS device is connected.	0x0
2	ENIRQHOSTDISCON	RW	Enable IRQ for Host disconnect: Enables interrupt for detection of disconnection to Device when in high-speed host mode. This should be enabled after ENDEVPLUGINDETECT is enabled.	0x0
3	HOSTDISCONDETECT_IRQ	RW	Device disconnect indication.: Indicates that the device has disconnected in High-Speed mode. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0
4	ENDEVPLUGINDET	RW	Enables non-standard resistive plugged-in detection.: This bit field controls connection of nominal 200kO resistors to both the USB_DP and USB_DM pins as one method of detecting when a USB cable is attached in device mode. This bit field must remain at a value of 1'b0 for normal USB data communication, or when using the USBHSDCD module for battery charger detection per the USB Battery Charger Specification Revision 1.2 or any other detection mechanism for USB cable plugin. The results of this detection method are reported in USBPHY_STATUS[6]. • 0 - Disables 200kO pull-up resistors on USB_DP and USB_DM pins (Default). • 1 - Enables 200kO pull-up resistors on USB_DP and USB_DM pins (Default).	0x0
5	DEVPLUGIN_POLARITY	RW	Device plugin polarity: For device mode, if this bit is cleared to 0, then it trips the interrupt if the device is plugged in. If set to 1, then it trips the interrupt if the device is unplugged.	0x0
7:6	-	RO	Reserved.	0x0
8	RESUMEIRQSTICKY	RW	Resume IRQ: Set to 1 will make RESUME_IRQ bit a sticky bit until software clear it. Set to 0, RESUME_IRQ only set during the wake-up period.	0x0
9	ENIRQRESUMEDETECT	RW	Enable IRQ Resume detect: Enables interrupt for detection of a non-J state on the USB line. This should only be enabled after the device has entered suspend mode.	0x0
10	RESUME_IRQ	RW	Resume IRQ: Indicates that the host is sending a wake-up after suspend. This bit is also set on a reset during suspend. Use this bit to wake up from suspend for either the resume or the reset case. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0
11	-	RW	Reserved.	0x0
12	DEVPLUGIN_IRQ	RW	Device connected indicator: Indicates that the device is connected. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0
13	-	RW	Reserved.	0x0

Table 863. General purpose control register (CTRL_TOG, offset = 0x3C) *…continued*

Bit	Symbol	Access	Description	Reset value
14	ENUTMILEVEL2	RW	Enable level 2 operation: Enables UTMI+ Level 2 operation for the USB HS PHY. This should be enabled if an Embedded Host use case needs to support a LS device.	0x0
15	ENUTMILEVEL3	RW	Enable level 3 operation: Enables UTMI+ Level 3 operation for the USB HS PHY. This should be enabled if an Embedded Host use case needs to support an external FS Hub with a LS device connected.	0x0
16	ENIRQWAKEUP	RW	Enable wake-up IRQ: Enables interrupt for the wake-up events.	0x0
17	WAKEUP_IRQ	RW	Wake-up IRQ: Indicates that there is a wak-eup event. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.	0x0
18	AUTORESUME_EN	RW	Enable auto resume: Enable the auto resume feature. When set, HW will use 32 kHz clock to send Resume to respond to the device remote wake-up (for host mode only). It's useful when PLL is off and reference clock is also powered down.	0x0
19	ENAUTOCLR_CLKGATE	RW	Auto clear clock gate.: Enables the feature to auto-clear the CLKGATE bit if there is wake-up event while USB is suspended. This should be enabled if needed to support auto wake-up without software interaction.	0x0
20	ENAUTOCLR_PHY_PWD	RW	Auto clear PWD register bits.: Enables the feature to auto-clear the PWD register bits in USBPHY_PWD if there is wake-up event while USB is suspended. This should be enabled if needed to support auto wake-up without software interaction.	0x0
21	ENDPDMCHG_WKUP	RW	Enable DP DM change wake-up: Not for customer use. This bit field must remain at value 1'b0. Enables the feature to wake-up USB if DP/DM is toggled when USB is suspended.	0x0
22	-	RW	Reserved.	0x0
23	ENVBUSCHG_WKUP	RW	Enable VBUS change wake-up: Enables the feature to wake-up USB if VBUS is toggled when USB is suspended.	0x0
24	-	RW	Reserved.	0x0
25	ENAUTOCLR_USBCLKGATE	RW	Enable auto-clear USB Clock gate: Enables the feature to auto-clear the USB0_CLKGATE/USB1_CLKGATE register bit in HW_DIGCTL_CTRL if there is wake-up event on USB0/USB1 while USB0/USB1 is suspended. This should be enabled if needs to support auto wake-up without S/W's interaction.	0x0
26	ENAUTOSET_USBCLKS	RW	Enable auto-set of USB clocks: Enables the feature to auto-clear the EN_USB_CLKS register bits in HW_CLKCTRL_PLL1CTRL0/HW_CLKCTRL_PLL1CTRL1 if there is wake-up event on USB0/USB1 while USB0/USB1 is suspended. This should be enabled if needs to support auto wake-up without S/W's interaction.	0x0
27	-	RO	Reserved.	0x0

Table 863. General purpose control register (CTRL_TOG, offset = 0x3C) *...continued*

Bit	Symbol	Access	Description	Reset value
28	HOST_FORCE_LS_SE0	RW	FS EOP low-speed timing: Forces the next FS packet that is transmitted to have a EOP with low-speed timing. This bit is used in host mode for the resume sequence. After the packet is transferred, this bit is cleared. The design can use this function to force the LS SE0 or use the USBPHY_CTRL_UTMI_SUSPENDM to trigger this event when leaving suspend. This bit is used in conjunction with USBPHY_DEBUG_HOST_RESUME_DEBUG.	0x0
29	UTMI_SUSPENDM	RO	UTMI suspend: Used by the PHY to indicate a powered-down state. If all the power-down bits in the USBPHY_PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1 when USB controller entering into Suspend mode. UTMI_SUSPENDM is negative logic, as required by the UTMI specification.	0x0
30	CLKGATE	RW	UTMI clock gate: Gate UTMI Clocks. Clear to 0 to run clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated. Note this bit can be auto-cleared if there is any wake-up event when USB is suspended while ENAUTOCLR_CLKGATE bit of USBPHY_CTRL is enabled.	0x1
31	SFTRST	RW	Software reset: Writing a 1 to this bit will soft-reset the USBPHY_PWD, USBPHY_TX, USBPHY_RX, and USBPHY_CTRL registers. Set to 0 to release the PHY from reset.	0x1

45.6.17 Status register

Table 864. Status register (STATUS, offset = 0x40)

Bit	Symbol	Access	Description	Reset value
0	OK_STATUS_3V	RO	Indicates the USB 3v power rails are in range.	0x0
2:1	-	RO	Reserved.	0x0
3	HOSTDISCONDETECT_STATUS	RO	Host disconnect status: Indicates at the local host (downstream) port that the remote device has disconnected while in High-Speed mode. • 0 - USB cable disconnect has not been detected at the local host. • 1 - USB cable disconnect has been detected at the local host.	0x0
5:4	-	RO	Reserved.	0x0
6	DEVPLUGIN_STATUS	RO	Status indicator for non-standard resistive plugged-in detection. Indicates that the device has been connected on the USB_DP and USB_DM lines using the nonstandard resistive plugged-in detection method controlled by USBPHY_CTRL[4]. When a USB cable attached to a remote host is attached to the local device, the 15kO host pull downs will override the high value resistors used in this detection method. • 0 - No attachment to a USB host is detected. • 1 - Cable attachment to a USB host is detected.	0x0
7	-	RO	Reserved.	0x0
8	-	RW	Reserved.	0x0

Table 864. Status register (STATUS, offset = 0x40) *…continued*

Bit	Symbol	Access	Description	Reset value
9	-	RO	Reserved.	0x0
10	RESUME_STATUS	RO	Resume status: Indicates that the host is sending a wake-up after suspend and has triggered an interrupt.	0x0
31:11	-	RO	Reserved.	0x0

45.6.18 PLL SIC register

Table 865. PLL SIC register (PLL_SIC, offset = 0xA0)

Bit	Symbol	Access	Description	Reset value
5:0	-	RW	Reserved.	0x0
6	PLL_EN_USB_CLKS	RW	PLL clock enable: Enables the USB clock from PLL to USB PHY.	0x0
11:7	-	RW	Reserved.	0x0
12	PLL_POWER	RW	Power PLL: Power up the USB PLL. The real PLL power up is also controlled by hardware. Hardware will power down PLL when USB is suspended and the device doesn't use it.	0x0
13	PLL_ENABLE	RW	PLL enable: Enables the clock output from the USB PLL. The real PLL output enable signal is also controlled by PLL power up control. Hardware will disable the PLL output before power down PLL, and enable the PLL output after power up PLL. The software only needs to set it when initializing the PLL.	0x1
15:14	-	RW	Reserved. Read value is undefined, only zero should be written.	0x0
16	-	RW	Reserved. User must set this bit to 0x0	0x1
18:17	-	RW	Reserved.	0x0
19	REFBIAS_PWD_SEL	RW	Reference bias power control: Reference bias power down select. • 0 - Selects PLL_POWER to control the reference bias. • 1 - Selects REFBIAS_PWD to control the reference bias.	0x0
20	REFBIAS_PWD	RW	Power down reference bias: This bit is only used when REFBIAS_PWD_SEL is set to 1.	0x1
21	PLL_REG_ENABLE	RW	Enable PLL regulator: This field controls the USB PLL regulator, set to enable the regulator. SW must set this bit 15 us before settiing PLL_POWER to avoid glitches on PLL output clock.	0x0
24:22	PLL_DIV_SEL	RW	PLL Divider value: This field controls the USB PLL feedback loop divider. Valid range for divider values: 54-108. Fout = Fin div_select/2.0.The USB PLL is designed to produce a 480MHz output clock. This bit field allows use of different frequency signals for the PLL reference clock input connected to the OSCCLK signal from the system oscillator. When override is enabled through USBPHY_TRIM_OVERRIDE_EN[0], the USB PLL will use this register value: • 000 - 32MHz input clock (Divide by 15). • 001 - 30MHz input clock (Divide by 16). • 010 - 24MHz input clock (Divide by 20) • 011 - Reserved. • 100 - 20MHz input clock (Divide by 24). • 101 - 19.2MHz input clock (Divide by 25). • 110 - 16MHz input clock (Divide by 30). • 111 - 12MHz input clock (Divide by 40).	0x3

Table 865. PLL SIC register (PLL_SIC, offset = 0xA0) *…continued*

Bit	Symbol	Access	Description	Reset value
26:25	-	RW	Reserved. User must set these bits to 0x0	0x1
29:27	-	RW	Reserved.	0x1
30	PLL_PREDIV	RW	This is selection between /1 or /2 to expand the range of ref input clock.	0x0
31	PLL_LOCK	RO	USB PLL lock status indicator 1 • 0 - PLL is not currently locked. • - PLL is currently locked.	0x0

45.6.19 PLL SIC register

Table 866. PLL SIC register (PLL_SIC_SET, offset = 0xA4)

Bit	Symbol	Access	Description	Reset value
5:0	-	RW	Reserved.	0x0
6	PLL_EN_USB_CLKS	RW	PLL clock enable: Enables the USB clock from PLL to USB PHY.	0x0
11:7	-	RW	Reserved.	0x0
12	PLL_POWER	RW	Power PLL: Power up the USB PLL. The real PLL power up is also controlled by hardware. Hardware will power down PLL when USB is suspended and the device doesn't use it.	0x0
13	PLL_ENABLE	RW	PLL enable: Enables the clock output from the USB PLL. The real PLL output enable signal is also controlled by PLL power up control. Hardware will disable the PLL output before power down PLL, and enable the PLL output after power up PLL. The software only needs to set it when initializing the PLL.	0x1
15:14	-	RW	Reserved. Read value is undefined, only zero should be written.	0x0
16	-	RW	Reserved. User must set this bit to 0x0	0x1
18:17	-	RW	Reserved.	0x0
19	REFBIAS_PWD_SEL	RW	Reference bias power control: Reference bias power down select. • 0 - Selects PLL_POWER to control the reference bias. • 1 - Selects REFBIAS_PWD to control the reference bias.	0x0
20	REFBIAS_PWD	RW	Power down reference bias: This bit is only used when REFBIAS_PWD_SEL is set to 1.	0x1
21	PLL_REG_ENABLE	RW	Enable PLL regulator: This field controls the USB PLL regulator, set to enable the regulator. SW must set this bit 15 us before settiing PLL_POWER to avoid glitches on PLL output clock.	0x0
24:22	PLL_DIV_SEL	RW	PLL Divider value: This field controls the USB PLL feedback loop divider. Valid range for divider values: 54-108. Fout = Fin div_select/2.0.The USB PLL is designed to produce a 480MHz output clock. This bit field allows use of different frequency signals for the PLL reference clock input connected to the OSCCLK signal from the system oscillator. When override is enabled through USBPHY_TRIM_OVERRIDE_EN[0], the USB PLL will use this register value. 000 - 32MHz input clock (Divide by 15) 001 - 30MHz input clock (Divide by 16) 010 - 24MHz input clock (Divide by 20) 011 - Reserved 100 - 20MHz input clock (Divide by 24) 101 - 19.2MHz input clock (Divide by 25) 110 - 16MHz input clock (Divide by 30).	0x3
26:25	-	RW	Reserved. User must set these bits to 0x0	0x1

Table 866. PLL SIC register (PLL_SIC_SET, offset = 0xA4) *…continued*

Bit	Symbol	Access	Description	Reset value
29:27	-	RW	Reserved.	0x1
30	PLL_PREDIV	RW	This is selection between /1 or /2 to expand the range of ref input clock.	0x0
31	PLL_LOCK	RO	USB PLL lock status indicator 1 • 0 - PLL is not currently locked. • - PLL is currently locked.	0x0

45.6.20 PLL SIC register

Table 867. PLL SIC register (PLL_SIC_CLR, offset = 0xA8)

Bit	Symbol	Access	Description	Reset value
5:0	-	RW	Reserved.	0x0
6	PLL_EN_USB_CLKS	RW	PLL clock enable: Enables the USB clock from PLL to USB PHY.	0x0
11:7	-	RW	Reserved.	0x0
12	PLL_POWER	RW	Power PLL: Power up the USB PLL. The real PLL power up is also controlled by hardware. Hardware will power down PLL when USB is suspended and the device doesn't use it.	0x0
13	PLL_ENABLE	RW	PLL enable: Enables the clock output from the USB PLL. The real PLL output enable signal is also controlled by PLL power up control. Hardware will disable the PLL output before power down PLL, and enable the PLL output after power up PLL. The software only needs to set it when initializing the PLL.	0x1
15:14	-	RW	Reserved. Read value is undefined, only zero should be written.	0x0
16	-	RW	Reserved. User must set this bit to 0x0	0x1
18:17	-	RW	Reserved.	0x0
19	REFBIAS_PWD_SEL	RW	Reference bias power control: Reference bias power down select. • 0 - Selects PLL_POWER to control the reference bias. • 1 - Selects REFBIAS_PWD to control the reference bias.	0x0
20	REFBIAS_PWD	RW	Power down reference bias: This bit is only used when REFBIAS_PWD_SEL is set to 1.	0x1
21	PLL_REG_ENABLE	RW	Enable PLL regulator: This field controls the USB PLL regulator, set to enable the regulator. SW must set this bit 15 us before settiing PLL_POWER to avoid glitches on PLL output clock.	0x0
24:22	PLL_DIV_SEL	RW	PLL Divider value: This field controls the USB PLL feedback loop divider. Valid range for divider values: 54-108. Fout = Fin div_select/2.0.The USB PLL is designed to produce a 480MHz output clock. This bit field allows use of different frequency signals for the PLL reference clock input connected to the OSCCLK signal from the system oscillator. When override is enabled through USBPHY_TRIM_OVERRIDE_EN[0], the USB PLL will use this register value. 000 - 32MHz input clock (Divide by 15) 001 - 30MHz input clock (Divide by 16) 010 - 24MHz input clock (Divide by 20) 011 - Reserved 100 - 20MHz input clock (Divide by 24) 101 - 19.2MHz input clock (Divide by 25) 110 - 16MHz input clock (Divide by 30).	0x3
26:25	-	RW	Reserved. User must set these bits to 0x0	0x1

Table 867. PLL SIC register (PLL_SIC_CLR, offset = 0xA8) *…continued*

Bit	Symbol	Access	Description	Reset value
29:27	-	RW	Reserved.	0x1
30	PLL_PREDIV	RW	This is selection between /1 or /2 to expand the range of ref input clock.	0x0
31	PLL_LOCK	RO	USB PLL lock status indicator 1 • 0 - PLL is not currently locked. • - PLL is currently locked.	0x0

45.6.21 PLL SIC register

Table 868. PLL SIC register (PLL_SIC_TOG, offset = 0xAC)

Bit	Symbol	Access	Description	Reset value
5:0	-	RW	Reserved.	0x0
6	PLL_EN_USB_CLKS	RW	PLL clock enable: Enables the USB clock from PLL to USB PHY.	0x0
11:7	-	RW	Reserved.	0x0
12	PLL_POWER	RW	Power PLL: Power up the USB PLL. The real PLL power up is also controlled by hardware. Hardware will power down PLL when USB is suspended and the device doesn't use it.	0x0
13	PLL_ENABLE	RW	PLL enable: Enables the clock output from the USB PLL. The real PLL output enable signal is also controlled by PLL power up control. Hardware will disable the PLL output before power down PLL, and enable the PLL output after power up PLL. The software only needs to set it when initializing the PLL.	0x1
15:14	-	RW	Reserved. Read value is undefined, only zero should be written.	0x0
16	-	RW	Reserved. User must set this bit to 0x0	0x1
18:17	-	RW	Reserved.	0x0
19	REFBIAS_PWD_SEL	RW	Reference bias power control: Reference bias power down select. • 0 - Selects PLL_POWER to control the reference bias. • 1 - Selects REFBIAS_PWD to control the reference bias.	0x0
20	REFBIAS_PWD	RW	Power down reference bias: This bit is only used when REFBIAS_PWD_SEL is set to 1.	0x1
21	PLL_REG_ENABLE	RW	Enable PLL regulator: This field controls the USB PLL regulator, set to enable the regulator. SW must set this bit 15 us before settiing PLL_POWER to avoid glitches on PLL output clock.	0x0
24:22	PLL_DIV_SEL	RW	PLL Divider value: This field controls the USB PLL feedback loop divider. Valid range for divider values: 54-108. Fout = Fin div_select/2.0.The USB PLL is designed to produce a 480MHz output clock. This bit field allows use of different frequency signals for the PLL reference clock input connected to the OSCCLK signal from the system oscillator. When override is enabled through USBPHY_TRIM_OVERRIDE_EN[0], the USB PLL will use this register value. 000 - 32MHz input clock (Divide by 15) 001 - 30MHz input clock (Divide by 16) 010 - 24MHz input clock (Divide by 20) 011 - Reserved 100 - 20MHz input clock (Divide by 24) 101 - 19.2MHz input clock (Divide by 25) 110 - 16MHz input clock (Divide by 30).	0x3
26:25	-	RW	Reserved. User must set these bits to 0x0	0x1

Table 868. PLL SIC register (PLL_SIC_TOG, offset = 0xAC) *…continued*

Bit	Symbol	Access	Description	Reset value
29:27	-	RW	Reserved.	0x1
30	PLL_PREDIV	RW	This is selection between /1 or /2 to expand the range of ref input clock.	0x0
31	PLL_LOCK	RO	USB PLL lock status indicator 1 • 0 - PLL is not currently locked. • - PLL is currently locked.	0x0

45.6.22 VBUS detect register

Table 869. VBUS detect register (USB1_VBUS_DETECT, offset = 0xC0)

Bit	Symbol	Access	Description	Reset value
2:0	VBUSVALID_THRESH	RW	VBUS comparator threshold: Sets the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5v, and includes hysteresis to minimize the need for software de-bounce of the detection. This comparator has 50mVof hystersis to prevent chattering at the comparator trip point. 000 - 4.0 V, 001 - 4.1 V, 010 - 4.2 V, 011 - 4.3 V ,100 - 4.4 V (Default), 101 - 4.5 V 110 - 4.6 V, 111 - 4.7 V	0x4
3	VBUS_OVERRIDE_EN	RW	VBUS detect signal override. This bit is used when EXT_VBUS_OVERRIDE_EN = 1'b0.: This bit field allows SW to override the results from the VBUS_VALID and Session Valid comparators using the values in USBPHY_USB1_VBUS_DETECT[7:4]. The VBUS_VALID, AVALID, BVALID, and SESSEND signals sent to the USB controller are each affected by these bit selections. The values reported for AVALID, BVALID, and SESSEND in USBPHY_USB1_VBUS_DET_STAT[2:0] are also affected but the value reported for VBUS_VALID in USBPHY_USB1_VBUS_DET_STAT[3] is not affected. This override method may be useful if VBUS detection is not done with the internal VBUS_VALID or Session End comparators. 0 - Use the results of the internal VBUS_VALID and Session Valid comparators for VBUS_VALID, AVALID, BVALID, and SESSEND (Default) 1 - Use the override values for VBUS_VALID, AVALID, BVALID, and SESSEND	0x0
4	SESSEND_OVERRIDE	RW	Override value for SESSEND: The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[0] if USBPHY_USB_VBUS_DETECT[3] is set to value 1'b1.	0x0
5	BVALID_OVERRIDE	RW	Override value for B-Device Session Valid: The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[1] if USBPHY_USB_VBUS_DETECT[3] is set to value 1'b1.	0x0
6	AVALID_OVERRIDE	RW	Override value for A-Device Session Valid: The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[2] if USBPHY_USB_VBUS_DETECT[3] is set to value 1'b1.	0x0

Table 869. VBUS detect register (USB1_VBUS_DETECT, offset = 0xC0) *…continued*

Bit	Symbol	Access	Description	Reset value
7	VBUSVALID_OVERRIDE	RW	Override value for VBUS_VALID signal sent to USB controller: The bit field provides the value for VBUS_VALID reported to the USB controller if the value of USBPHY_USB1_VBUS_DETECT[3] is set to 1'b1. The value of this bit field does not affect the value of USBPHY_USB1_VBUS_DET_STAT[3].	0x0
8	VBUSVALID_SEL	RW	Selects the source of the VBUS_VALID signal reported to the USB controller: This is one of the bit fields that selects the source of the VBUS_VALID signal reported to the USB controller. The VBUS_VALID source selection in this bit field only takes effect if USBPHY_USB1_VBUS_DETECT[3] has the value 1'b0. This bit field does not impact the VBUS_VALID value reported in USBPHY_USB1_VBUS_DET_STAT[3]. 0 - Use the VBUS_VALID comparator results for signal reported to the USB controller (Default) 1 - Use the VBUS_VALID_3V detector results for signal reported to the USB controller	0x0
10:9	VBUS_SOURCE_SEL	RW	Selects the source of the VBUS_VALID signal reported to the USB controller: This is one of the bit fields that selects the source of the VBUS_VALID signal reported to the USB controller. The VBUS_VALID source selections in this bit field only take effect if both USBPHY_USB1_VBUS_DETECT[8] and USBPHY_USB1_VBUS_DETECT[3] each have the value 1'b0. This bit field does not impact the VBUS_VALID value reported in USBPHY_USB1_VBUS_DET_STAT[3]. • 00 - Use the VBUS_VALID comparator results for signal reported to the USB controller (Default). • 01 - Use the Session Valid comparator results for signal reported to the USB controller. • 10 - Use the Session Valid comparator results for signal reported to the USB controller. • 11 - Reserved, do not use.	0x0
11	ID_OVERRIDE_EN	RW	Enable ID override using the register field. This bit is only used if EXT_ID_OVERRIDE_EN = 1'b0.	0x0
12	ID_OVERRIDE	RW	ID override value.	0x0
13	EXT_ID_OVERRIDE_EN	RW	Enable ID override using the pinmuxed value: 0 - Select the Muxed value chosen using ID_OVERRIDE_EN. 1 - Select the external ID value.	0x0
14	EXT_VBUS_OVERRIDE_EN	RW	Enable VBUS override using the pinmuxed value. • 0 - Select the Muxed value chosen using VBUS_OVERRIDE_EN. • 1 - Select the external VBUS VALID value.	0x0
17:15		RO	Reserved.	0x0

Table 869. VBUS detect register (USB1_VBUS_DETECT, offset = 0xC0) *…continued*

Bit	Symbol	Access	Description	Reset value
18	VBUSVALID_TO_SESSVALID	RW	Selects the comparator used for VBUS_VALID: This bit field controls the comparator used to report the VBUS_VALID results in USBPHY_USB1_VBUS_DETECT[3] between the VBUS_VALID comparator and the Session Valid comparator. The VBUS_VALID comparator is the most accurate and has a programmable threshold set by USBPHY_USB_VBUS_DETECT[2:0]. The Session Valid comparator may be useful in systems using nonstandard VBUS voltages. The mux selection in this bit field happens before any VBUS_VALID selection controlled by the USBPHY_USB1_VBUS_DETECT[10:8] bits. 0 - Use the VBUS_VALID comparator for VBUS_VALID results 1 - Use the Session End comparator for VBUS_VALID results. The Session End threshold is -> 0.8V and -< 4.0V.	0x0
19	-	RW	Reserved.	0x0
22:20	PWRUP_CMPS	RW	Enables the VBUS_VALID comparator: Powers up the comparator used for the VBUS_VALID detector. This bit field can be reset to value 3'h0 to save power if the internal VBUS_VALID comparator is not used. • 000 - Powers down the VBUS_VALID comparator. • 111 - Enables the VBUS_VALID comparator (default).	0x7
25:23		RO	Reserved.	0x0
26	DISCHARGE_VBUS	RW	Controls VBUS discharge resistor: This bit field controls a nominal 22kO resistor between the USB1_VBUS pin and ground. It can be used to accelerate the fall of the VBUS signal at the end of a session. • 0 - VBUS discharge resistor is disabled (Default). • 1 - VBUS discharge resistor is enabled.	0x0
31:27	-	RW	Reserved.	0x0

45.6.23 VBUS detect register set

Table 870. VBUS detect Register (USB1_VBUS_DETECT_SET, offset = 0xC4)

Bit	Symbol	Access	Description	Reset value
2:0	VBUSVALID_THRESH	RW	VBUS comparator threshold: Sets the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5v, and includes hystersis to minimize the need for software de-bounce of the detection. This comparator has 50mVof hystersis to prevent chattering at the comparator trip point. 000 - 4.0 V 001 - 4.1 V 010 - 4.2 V 011 - 4.3 V 100 - 4.4 V (Default) 101 - 4.5 V 110 - 4.6 V 111 - 4.7 V	0x4
3	VBUS_OVERRIDE_EN	RW	VBUS detect signal override. This bit is used when EXT_VBUS_OVERRIDE_EN = 1’b0.: This bit field allows SW to override the results from the VBUS_VALID and Session Valid comparators using the values in USBPHY_USB1_VBUS_DETECT[7:4]. The VBUS_VALID, AVALID, BVALID, and SESSEND signals sent to the USB controller are each affected by these bit selections. The values reported for AVALID, BVALID, and SESSEND in USBPHY_USB1_VBUS_DET_STAT[2:0] are also affected but the value reported for VBUS_VALID in USBPHY_USB1_VBUS_DET_STAT[3] is not affected. This override method may be useful if VBUS detection is not done with the internal VBUS_VALID or Session End comparators. 0 - Use the results of the internal VBUS_VALID and Session Valid comparators for VBUS_VALID, AVALID, BVALID, and SESSEND (Default) 1 - Use the override values for VBUS_VALID, AVALID, BVALID, and SESSEND	0x0
4	SESSEND_OVERRIDE	RW	Override value for SESSEND: The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[0] if USBPHY_USB_VBUS_DETECT[3] is set to value 1’b1.	0x0
5	BVALID_OVERRIDE	RW	Override value for B-Device Session Valid: The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[1] if USBPHY_USB_VBUS_DETECT[3] is set to value 1’b1.	0x0
6	AVALID_OVERRIDE	RW	Override value for A-Device Session Valid: The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[2] if USBPHY_USB_VBUS_DETECT[3] is set to value 1’b1.	0x0
7	VBUSVALID_OVERRIDE	RW	Override value for VBUS_VALID signal sent to USB controller: The bit field provides the value for VBUS_VALID reported to the USB controller if the value of USBPHY_USB1_VBUS_DETECT[3] is set to 1’b1. The value of this bit field does not affect the value of USBPHY_USB1_VBUS_DET_STAT[3].	0x0

Table 870. VBUS detect Register (USB1_VBUS_DETECT_SET, offset = 0xC4) *…continued*

Bit	Symbol	Access	Description	Reset value
8	VBUSVALID_SEL	RW	Selects the source of the VBUS_VALID signal reported to the USB controller: This is one of the bit fields that selects the source of the VBUS_VALID signal reported to the USB controller. The VBUS_VALID source selection in this bit field only takes effect if USBPHY_USB1_VBUS_DETECT[3] has the value 1'b0. This bit field does not impact the VBUS_VALID value reported in USBPHY_USB1_VBUS_DET_STAT[3]. 0 - Use the VBUS_VALID comparator results for signal reported to the USB controller (Default) 1 - Use the VBUS_VALID_3V detector results for signal reported to the USB controller	0x0
10:9	VBUS_SOURCE_SEL	RW	Selects the source of the VBUS_VALID signal reported to the USB controller: This is one of the bit fields that selects the source of the VBUS_VALID signal reported to the USB controller. The VBUS_VALID source selections in this bit field only take effect if both USBPHY_USB1_VBUS_DETECT[8] and USBPHY_USB1_VBUS_DETECT[3] each have the value 1'b0. This bit field does not impact the VBUS_VALID value reported in USBPHY_USB1_VBUS_DET_STAT[3]. • 00 - Use the VBUS_VALID comparator results for signal reported to the USB controller (Default). • 01 - Use the Session Valid comparator results for signal reported to the USB controller. • 10 - Use the Session Valid comparator results for signal reported to the USB controller. • 11 - Reserved, do not use.	0x0
11	ID_OVERRIDE_EN	RW	Enable ID override using the register field. This bit is only used if EXT_ID_OVERRIDE_EN = 1'b0.	0x0
12	ID_OVERRIDE	RW	ID override value.	0x0
13	EXT_ID_OVERRIDE_EN	RW	Enable ID override using the pinmuxed value: 0 - Select the Muxed value chosen using ID_OVERRIDE_EN. 1 - Select the external ID value.	0x0
14	EXT_VBUS_OVERRIDE_EN	RW	Enable VBUS override using the pinmuxed value. • 0 - Select the Muxed value chosen using VBUS_OVERRIDE_EN. • 1 - Select the external VBUS VALID value.	0x0
17:15		RO	Reserved.	0x0

Table 870. VBUS detect Register (USB1_VBUS_DETECT_SET, offset = 0xC4) *…continued*

Bit	Symbol	Access	Description	Reset value
18	VBUSVALID_TO_SESSVALID	RW	Selects the comparator used for VBUS_VALID: This bit field controls the comparator used to report the VBUS_VALID results in USBPHY_USB1_VBUS_DETECT[3] between the VBUS_VALID comparator and the Session Valid comparator. The VBUS_VALID comparator is the most accurate and has a programmable threshold set by USBPHY_USB_VBUS_DETECT[2:0]. The Session Valid comparator may be useful in systems using nonstandard VBUS voltages. The mux selection in this bit field happens before any VBUS_VALID selection controlled by the USBPHY_USB1_VBUS_DETECT[10:8] bits. 0 - Use the VBUS_VALID comparator for VBUS_VALID results 1 - Use the Session End comparator for VBUS_VALID results. The Session End threshold is -> 0.8V and -< 4.0V.	0x0
19	-	RW	Reserved.	0x0
22:20	PWRUP_CMPS	RW	Enables the VBUS_VALID comparator: Powers up the comparator used for the VBUS_VALID detector. This bit field can be reset to value 3'h0 to save power if the internal VBUS_VALID comparator is not used. • 000 - Powers down the VBUS_VALID comparator. • 111 - Enables the VBUS_VALID comparator (default).	0x7
25:23		RO	Reserved.	0x0
26	DISCHARGE_VBUS	RW	Controls VBUS discharge resistor: This bit field controls a nominal 22kO resistor between the USB1_VBUS pin and ground. It can be used to accelerate the fall of the VBUS signal at the end of a session. • 0 - VBUS discharge resistor is disabled (Default). • 1 - VBUS discharge resistor is enabled.	0x0
30:27		RO	Reserved.	0x0
31:28	-	RW	Reserved.	0x0

45.6.24 VBUS detect register Clear

Table 871. VBUS detect Register (USB1_VBUS_DETECT_CLR, offset = 0xC8)

Bit	Symbol	Access	Description	Reset value
2:0	VBUSVALID_THRESH	RW	VBUS comparator threshold: Sets the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5v, and includes hystersis to minimize the need for software de-bounce of the detection. This comparator has 50mVof hystersis to prevent chattering at the comparator trip point. 000 - 4.0 V 001 - 4.1 V 010 - 4.2 V 011 - 4.3 V 100 - 4.4 V (Default) 101 - 4.5 V 110 - 4.6 V 111 - 4.7 V	0x4
3	VBUS_OVERRIDE_EN	RW	VBUS detect signal override. This bit is used when EXT_VBUS_OVERRIDE_EN = 1’b0.: This bit field allows SW to override the results from the VBUS_VALID and Session Valid comparators using the values in USBPHY_USB1_VBUS_DETECT[7:4]. The VBUS_VALID, AVALID, BVALID, and SESSEND signals sent to the USB controller are each affected by these bit selections. The values reported for AVALID, BVALID, and SESSEND in USBPHY_USB1_VBUS_DET_STAT[2:0] are also affected but the value reported for VBUS_VALID in USBPHY_USB1_VBUS_DET_STAT[3] is not affected. This override method may be useful if VBUS detection is not done with the internal VBUS_VALID or Session End comparators. 0 - Use the results of the internal VBUS_VALID and Session Valid comparators for VBUS_VALID, AVALID, BVALID, and SESSEND (Default) 1 - Use the override values for VBUS_VALID, AVALID, BVALID, and SESSEND	0x0
4	SESSEND_OVERRIDE	RW	Override value for SESSEND: The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[0] if USBPHY_USB_VBUS_DETECT[3] is set to value 1’b1.	0x0
5	BVALID_OVERRIDE	RW	Override value for B-Device Session Valid: The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[1] if USBPHY_USB_VBUS_DETECT[3] is set to value 1’b1.	0x0
6	AVALID_OVERRIDE	RW	Override value for A-Device Session Valid: The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[2] if USBPHY_USB_VBUS_DETECT[3] is set to value 1’b1.	0x0
7	VBUSVALID_OVERRIDE	RW	Override value for VBUS_VALID signal sent to USB controller: The bit field provides the value for VBUS_VALID reported to the USB controller if the value of USBPHY_USB1_VBUS_DETECT[3] is set to 1’b1. The value of this bit field does not affect the value of USBPHY_USB1_VBUS_DET_STAT[3].	0x0

Table 871. VBUS detect Register (USB1_VBUS_DETECT_CLR, offset = 0xC8) *…continued*

Bit	Symbol	Access	Description	Reset value
8	VBUSVALID_SEL	RW	Selects the source of the VBUS_VALID signal reported to the USB controller: This is one of the bit fields that selects the source of the VBUS_VALID signal reported to the USB controller. The VBUS_VALID source selection in this bit field only takes effect if USBPHY_USB1_VBUS_DETECT[3] has the value 1'b0. This bit field does not impact the VBUS_VALID value reported in USBPHY_USB1_VBUS_DET_STAT[3]. 0 - Use the VBUS_VALID comparator results for signal reported to the USB controller (Default) 1 - Use the VBUS_VALID_3V detector results for signal reported to the USB controller	0x0
10:9	VBUS_SOURCE_SEL	RW	Selects the source of the VBUS_VALID signal reported to the USB controller: This is one of the bit fields that selects the source of the VBUS_VALID signal reported to the USB controller. The VBUS_VALID source selections in this bit field only take effect if both USBPHY_USB1_VBUS_DETECT[8] and USBPHY_USB1_VBUS_DETECT[3] each have the value 1'b0. This bit field does not impact the VBUS_VALID value reported in USBPHY_USB1_VBUS_DET_STAT[3]. • 00 - Use the VBUS_VALID comparator results for signal reported to the USB controller (Default). • 01 - Use the Session Valid comparator results for signal reported to the USB controller. • 10 - Use the Session Valid comparator results for signal reported to the USB controller. • 11 - Reserved, do not use.	0x0
11	ID_OVERRIDE_EN	RW	Enable ID override using the register field. This bit is only used if EXT_ID_OVERRIDE_EN = 1'b0.	0x0
12	ID_OVERRIDE	RW	ID override value.	0x0
13	EXT_ID_OVERRIDE_EN	RW	Enable ID override using the pinmuxed value: 0 - Select the Muxed value chosen using ID_OVERRIDE_EN. 1 - Select the external ID value.	0x0
14	EXT_VBUS_OVERRIDE_EN	RW	Enable VBUS override using the pin muxed value. • 0 - Select the muxed value chosen using VBUS_OVERRIDE_EN. • 1 - Select the external VBUS VALID value.	0x0
17:15		RO	Reserved.	0x0

Table 871. VBUS detect Register (USB1_VBUS_DETECT_CLR, offset = 0xC8) *…continued*

Bit	Symbol	Access	Description	Reset value
18	VBUSVALID_TO_SESSVALID	RW	Selects the comparator used for VBUS_VALID: This bit field controls the comparator used to report the VBUS_VALID results in USBPHY_USB1_VBUS_DETECT[3] between the VBUS_VALID comparator and the Session Valid comparator. The VBUS_VALID comparator is the most accurate and has a programmable threshold set by USBPHY_USB_VBUS_DETECT[2:0]. The Session Valid comparator may be useful in systems using nonstandard VBUS voltages. The mux selection in this bit field happens before any VBUS_VALID selection controlled by the USBPHY_USB1_VBUS_DETECT[10:8] bits. 0 - Use the VBUS_VALID comparator for VBUS_VALID results 1 - Use the Session End comparator for VBUS_VALID results. The Session End threshold is -> 0.8V and -< 4.0V.	0x0
19	-	RW	Reserved.	0x0
22:20	PWRUP_CMPS	RW	Enables the VBUS_VALID comparator: Powers up the comparator used for the VBUS_VALID detector. This bit field can be reset to value 3'h0 to save power if the internal VBUS_VALID comparator is not used. • 000 - Powers down the VBUS_VALID comparator. • 111 - Enables the VBUS_VALID comparator (default).	0x7
25:23		RO	Reserved.	0x0
26	DISCHARGE_VBUS	RW	Controls VBUS discharge resistor: This bit field controls a nominal 22kO resistor between the USB1_VBUS pin and ground. It can be used to accelerate the fall of the VBUS signal at the end of a session. • 0 - VBUS discharge resistor is disabled (Default). • 1 - VBUS discharge resistor is enabled.	0x0
31:27	-	RW	Reserved.	0x0

45.6.25 VBUS detect register Toggle

Table 872. VBUS detect Register (USB1_VBUS_DETECT_TOG, offset = 0xCC)

Bit	Symbol	Access	Description	Reset value
2:0	VBUSVALID_THRESH	RW	VBUS comparator threshold: Sets the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5 V, and includes hystersis to minimize the need for software de-bounce of the detection. This comparator has 50mVof hystersis to prevent chattering at the comparator trip point. 000 - 4.0 V 001 - 4.1 V 010 - 4.2 V 011 - 4.3 V 100 - 4.4 V (Default) 101 - 4.5 V 110 - 4.6 V 111 - 4.7 V	0x4
3	VBUS_OVERRIDE_EN	RW	VBUS detect signal override. This bit is used when EXT_VBUS_OVERRIDE_EN = 1'b0.: This bit field allows SW to override the results from the VBUS_VALID and Session Valid comparators using the values in USBPHY_USB1_VBUS_DETECT[7:4]. The VBUS_VALID, AVALID, BVALID, and SESSEND signals sent to the USB controller are each affected by these bit selections. The values reported for AVALID, BVALID, and SESSEND in USBPHY_USB1_VBUS_DET_STAT[2:0] are also affected but the value reported for VBUS_VALID in USBPHY_USB1_VBUS_DET_STAT[3] is not affected. This override method may be useful if VBUS detection is not done with the internal VBUS_VALID or Session End comparators. • 0 - Use the results of the internal VBUS_VALID and Session Valid comparators for VBUS_VALID, AVALID, BVALID, and SESSEND (Default). • 1 - Use the override values for VBUS_VALID, AVALID, BVALID, and SESSEND.	0x0
4	SESSEND_OVERRIDE	RW	Override value for SESSEND: The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[0] if USBPHY_USB_VBUS_DETECT[3] is set to value 1'b1.	0x0
5	BVALID_OVERRIDE	RW	Override value for B-Device Session Valid: The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[1] if USBPHY_USB_VBUS_DETECT[3] is set to value 1'b1.	0x0
6	AVALID_OVERRIDE	RW	Override value for A-Device Session Valid: The bit field provides the value for USBPHY_USB1_VBUS_DET_STAT[2] if USBPHY_USB_VBUS_DETECT[3] is set to value 1'b1.	0x0
7	VBUSVALID_OVERRIDE	RW	Override value for VBUS_VALID signal sent to USB controller: The bit field provides the value for VBUS_VALID reported to the USB controller if the value of USBPHY_USB1_VBUS_DETECT[3] is set to 1'b1. The value of this bit field does not affect the value of USBPHY_USB1_VBUS_DET_STAT[3].	0x0

Table 872. VBUS detect Register (USB1_VBUS_DETECT_TOG, offset = 0xCC) *…continued*

Bit	Symbol	Access	Description	Reset value
8	VBUSVALID_SEL	RW	Selects the source of the VBUS_VALID signal reported to the USB controller: This is one of the bit fields that selects the source of the VBUS_VALID signal reported to the USB controller. The VBUS_VALID source selection in this bit field only takes effect if USBPHY_USB1_VBUS_DETECT[3] has the value 1'b0. This bit field does not impact the VBUS_VALID value reported in USBPHY_USB1_VBUS_DET_STAT[3]. • 0 - Use the VBUS_VALID comparator results for signal reported to the USB controller (Default). • 1 - Use the VBUS_VALID_3V detector results for signal reported to the USB controller.	0x0
10:9	VBUS_SOURCE_SEL	RW	Selects the source of the VBUS_VALID signal reported to the USB controller: This is one of the bit fields that selects the source of the VBUS_VALID signal reported to the USB controller. The VBUS_VALID source selections in this bit field only take effect if both USBPHY_USB1_VBUS_DETECT[8] and USBPHY_USB1_VBUS_DETECT[3] each have the value 1'b0. This bit field does not impact the VBUS_VALID value reported in USBPHY_USB1_VBUS_DET_STAT[3]. • 00 - Use the VBUS_VALID comparator results for signal reported to the USB controller (Default). • 01 - Use the Session Valid comparator results for signal reported to the USB controller. • 10 - Use the Session Valid comparator results for signal reported to the USB controller. • 11 - Reserved, do not use.	0x0
11	ID_OVERRIDE_EN	RW	Enable ID override using the register field. This bit is only used if EXT_ID_OVERRIDE_EN = 1'b0.	0x0
12	ID_OVERRIDE	RW	ID override value.	0x0
13	EXT_ID_OVERRIDE_EN	RW	Enable ID override using the pin muxed value. • 0 - Select the muxed value chosen using ID_OVERRIDE_EN. • 1 - Select the external ID value.	0x0
14	EXT_VBUS_OVERRIDE_EN	RW	Enable VBUS override using the pin muxed value. • 0 - Select the Muxed value chosen using VBUS_OVERRIDE_EN. • 1 - Select the external VBUS VALID value.	0x0
17:15		RO	Reserved.	0x0

Table 872. VBUS detect Register (USB1_VBUS_DETECT_TOG, offset = 0xCC) *…continued*

Bit	Symbol	Access	Description	Reset value
18	VBUSVALID_TO_SESSVALID	RW	Selects the comparator used for VBUS_VALID: This bit field controls the comparator used to report the VBUS_VALID results in USBPHY_USB1_VBUS_DETECT[3] between the VBUS_VALID comparator and the Session Valid comparator. The VBUS_VALID comparator is the most accurate and has a programmable threshold set by USBPHY_USB_VBUS_DETECT[2:0]. The Session Valid comparator may be useful in systems using nonstandard VBUS voltages. The mux selection in this bit field happens before any VBUS_VALID selection controlled by the USBPHY_USB1_VBUS_DETECT[10:8] bits. • 0 - Use the VBUS_VALID comparator for VBUS_VALID results. • 1 - Use the Session End comparator for VBUS_VALID results. The Session End threshold is -> 0.8V and -< 4.0V.	0x0
19	VBUSVALID_5VDETECT	RW		0x0
22:20	PWRUP_CMPS	RW	Enables the VBUS_VALID comparator: Powers up the comparator used for the VBUS_VALID detector. This bit field can be reset to value 3'h0 to save power if the internal VBUS_VALID comparator is not used. • 000 - Powers down the VBUS_VALID comparator. • 111 - Enables the VBUS_VALID comparator (default).	0x7
25:23		RO	Reserved.	0x0
26	DISCHARGE_VBUS	RW	Controls VBUS discharge resistor: This bit field controls a nominal 22kO resistor between the USB1_VBUS pin and ground. It can be used to accelerate the fall of the VBUS signal at the end of a session. • 0 - VBUS discharge resistor is disabled (Default). • 1 - VBUS discharge resistor is enabled.	0x0
31:27	-	RW	Reserved.	0x0

45.6.26 VBUS detect register Status

Table 873. VBUS detect Register (USB1_VBUS_DETECT_TOG, offset = 0xCC)

Bit	Symbol	Access	Description	Reset value
0	SESSEND	RO	SESSEND. • 0 - Use the results of the internal VBUS_VALID and Session Valid comparators for VBUS_VAL, BVALID, and SESSEND (Default). • 1 - The VBUS voltage is below the Session Valid threshold.	0x0
1	BVALID	RO	BVALID. • 0 - The VBUS voltage is below the Session Valid threshold. • 1 - The VBUS voltage is above the Session Valid threshold.	0x0
2	AVALID	RO	AVALID. • 0 - The VBUS voltage is below the Session Valid threshold. • 1 - The VBUS voltage is above the Session Valid threshold.	0x0
3	VBUS_VALID	RO	VBUS_VALID. • 0 - VBUS is below the comparator threshold. • 1 - VBUS is above the comparator threshold.	0x0
4	VBUS_VALID_3V	RO	VBUS_VALID. • 0 - VBUS voltage is below VBUS_VALID_3V threshold. • 1 - VBUS voltage is above VBUS_VALID_3V threshold.	0x0
31:5	-	-	Reserved.	0x0

45.6.27 Analog control register

Table 874. Analog Control Register (ANACTRL, offset = 0x100)

Bit	Symbol	Access	Description	Reset value
0	-	RW	Reserved. Only zero should be written.	0x0
1	LVI_EN	RW	LVI EN.: Vow voltage detector enable bit.	0x0
3:2	PFD_CLK_SEL	RW	For normal USB operation, this bit field must remain at value 2’b00.	0x0
9:4	-	RW	Reserved. Only zero should be written.	0x0
10	DEV_PULLDOWN	RW	Setting this field to 1’b1 will enable the 15kO pull down resistors on both USB_DP and USB_DM pins. This feature can be used in device mode while the USB cable is disconnected to keep the data pins at known values, avoiding unnecessary interrupts from the single ended receivers. This bit must be reset to 1’b0 during normal USB data communication in device mode, or while battery charger detection using the USBHSDCD module is used. • 0 - The 15kO nominal pull downs on the USB_DP and USB_DM pins are disabled in device mode. • 1 - The 15kO nominal pull downs on the USB_DP and USB_DM pins are enabled in device mode.	0x1
31:11	-	RW	Reserved	0x0

45.6.28 Analog control register

Table 875. Analog Control Register (ANACTRL_SET, offset = 0x104)

Bit	Symbol	Access	Description	Reset value
0	-	RW	Reserved. Only zero should be written.	0x0
1	LVI_EN	RW	LVI EN.: Vow voltage detector enable bit.	0x0
3:2	PFD_CLK_SEL	RW	For normal USB operation, this bit field must remain at value 2’b00.	0x0
9:4	-	RW	Reserved. Only zero should be written.	0x0
10	DEV_PULLDOWN	RW	Setting this field to 1’b1 will enable the 15kO pull down resistors on both USB_DP and USB_DM pins. This feature can be used in device mode while the USB cable is disconnected to keep the data pins at known values, avoiding unnecessary interrupts from the single ended receivers. This bit must be reset to 1’b0 during normal USB data communication in device mode, or while battery charger detection using the USBHSDCD module is used. • 0 - The 15kO nominal pull downs on the USB_DP and USB_DM pins are disabled in device mode. • 1 - The 15kO nominal pull downs on the USB_DP and USB_DM pins are enabled in device mode.	0x1
31:11	-	RW	Reserved	0x0

45.6.29 Analog control register

Table 876. Analog Control Register (ANACTRL_CLR, offset = 0x108)

Bit	Symbol	Access	Description	Reset value
0	-	RW	Reserved. Only zero should be written.	0x0
1	LVI_EN	RW	LVI EN.: Vow voltage detector enable bit.	0x0
3:2	PFD_CLK_SEL	RW	For normal USB operation, this bit field must remain at value 2’b00.	0x0
9:4	-	RW	Reserved. Only zero should be written.	0x0
10	DEV_PULLDOWN	RW	Setting this field to 1’b1 will enable the 15kO pull down resistors on both USB_DP and USB_DM pins. This feature can be used in device mode while the USB cable is disconnected to keep the data pins at known values, avoiding unnecessary interrupts from the single ended receivers. This bit must be reset to 1’b0 during normal USB data communication in device mode, or while battery charger detection using the USBHSDCD module is used. • 0 - The 15kO nominal pull downs on the USB_DP and USB_DM pins are disabled in device mode. • 1 - The 15kO nominal pull downs on the USB_DP and USB_DM pins are enabled in device mode.	0x1
31:11	-	RW	Reserved	0x0

45.6.30 Analog control register

Table 877. Analog Control Register (ANACTRL_TOG, offset = 0x10C)

Bit	Symbol	Access	Description	Reset value
0	-	RW	Reserved. Only zero should be written.	0x0
1	LVI_EN	RW	LVI EN.: Vow voltage detector enable bit.	0x0
3:2	PFD_CLK_SEL	RW	For normal USB operation, this bit field must remain at value 2’b00.	0x0

Table 877. Analog Control Register (ANACTRL_TOG, offset = 0x10C) *…continued*

Bit	Symbol	Access	Description	Reset value
9:4	-	RW	Reserved. Only zero should be written.	0x0
10	DEV_PULLDOWN	RW	Setting this field to 1’b1 will enable the 15kO pull down resistors on both USB_DP and USB_DM pins. This feature can be used in device mode while the USB cable is disconnected to keep the data pins at known values, avoiding unnecessary interrupts from the single ended receivers. This bit must be reset to 1’b0 during normal USB data communication in device mode, or while battery charger detection using the USBHSDCD module is used. • 0 - The 15kO nominal pull downs on the USB_DP and USB_DM pins are disabled in device mode. • 1 - The 15kO nominal pull downs on the USB_DP and USB_DM pins are enabled in device mode.	0x1
31:11	-	RW	Reserved	0x0

46.1 How to read this chapter

The Cyclic Redundancy Check (CRC) engine is available on LPC55S6x/LPC55S2x/LPC552x devices.

46.2 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.
- Supports CPU PIO back-to-back transfer.
- Accepts variable size 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).

46.3 Basic configuration

Set the CRC bit in the AHBCLKCTRL0 register. See Section 4.5.17 "AHB clock control 0" to enable the clock for the CRC engine.

46.4 Pin description

The CRC engine has no configurable pins.

46.5 General description

The CRC generator, (with programmable polynomial settings), supports several commonly used CRC standards.

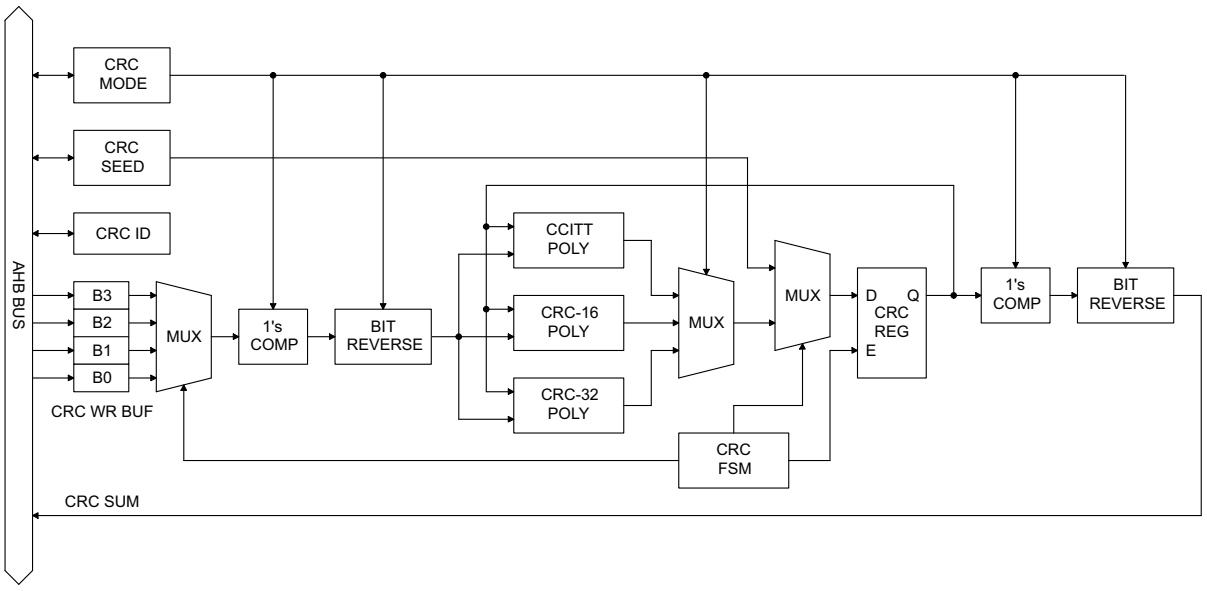


Fig 167. CRC block diagram

46.6 Register description

Table 878. Register overview: CRC engine (base address = 0x4009 5000)

Name	Access	Offset	Description	Reset value	Section
MODE	R/W	0x000	CRC mode register.	0x0000 0000	46.6.1
SEED	R/W	0x004	CRC seed register.	0x0000 FFFF	46.6.2
SUM	RO	0x008	CRC checksum register.	0x0000 FFFF	46.6.3
WR_DATA	WO	0x008	CRC data register.	-	46.6.4

46.6.1 CRC mode register

Table 879. CRC mode register (MODE, offset = 0x000)

Bit	Symbol	Description	Reset value
1:0	CRC_POLY	CRC polynomial: 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

Table 879. CRC mode register (MODE, offset = 0x000) *…continued*

Bit	Symbol	Description	Reset value
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

46.6.2 CRC seed register

Table 880. CRC seed register (SEED, offset = 0x004)

Bit	Symbol	Description	Reset value
31:0	CRC_SEED	A write access to this register will load a CRC seed value to the CRC_SUM register with a selected bit order and 1's complement pre-processes. **Remark:** A write access to this register will overrule the current CRC calculation in progresses.	0x0000 FFFF

46.6.3 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 881. CRC checksum register (SUM, offset = 0x008)

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

46.6.4 CRC data register

This register is a write-only register containing the data block for which the CRC sum will be calculated.

Table 882. CRC data register (WR_DATA, offset = 0x008)

Bit	Symbol	Description	Reset value
31:0	CRC_WR_DATA	Data written to this register will be used to perform a CRC calculation with the selected bit order and 1's complement pre-process. Any write size (8, 16 or 32-bit) is allowed and can include back-to-back transactions.	-

46.7 Functional description

The following sections describe the register settings for each supported CRC standard:

46.7.1 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

46.7.2 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

46.7.3 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

47.1 How to read this chapter

TrustZone for Armv8-M and the trusted execution environment is available on the LPC55S6x device. This section discusses the basic features offered by TrustZone for Armv8-M and then explains the additional features of the LPC55S6x that extends the TrustZone security foundation to enable a trusted execution environment.

47.2 Features

- CPU0 is Cortex-M33 with TrustZone support enabled.
- Attribution Units (SAU, IDAU)
- Secure MPU, secure NVIC, secure SYSTICK, secure Stack Pointer.
- Secure memory map aliasing.
- Support for ARM AMBA 5.0 AHB secure bus.
- Secure bus controller.
- Memory and peripheral protection checkers.
- Security attribution wrapper for AHB masters.
- Secure DMA and DMA masking.
- Secure GPIO and GPIO masking.
- Secure debug.

Remark: The AHB Secure Controller must be configured and its security check needs to be enabled for proper TrustZone functionality. See Section 47.3.5 “TrustZone configuration example” for details.

47.3 Functional description

47.3.1 TrustZone for Armv8-M

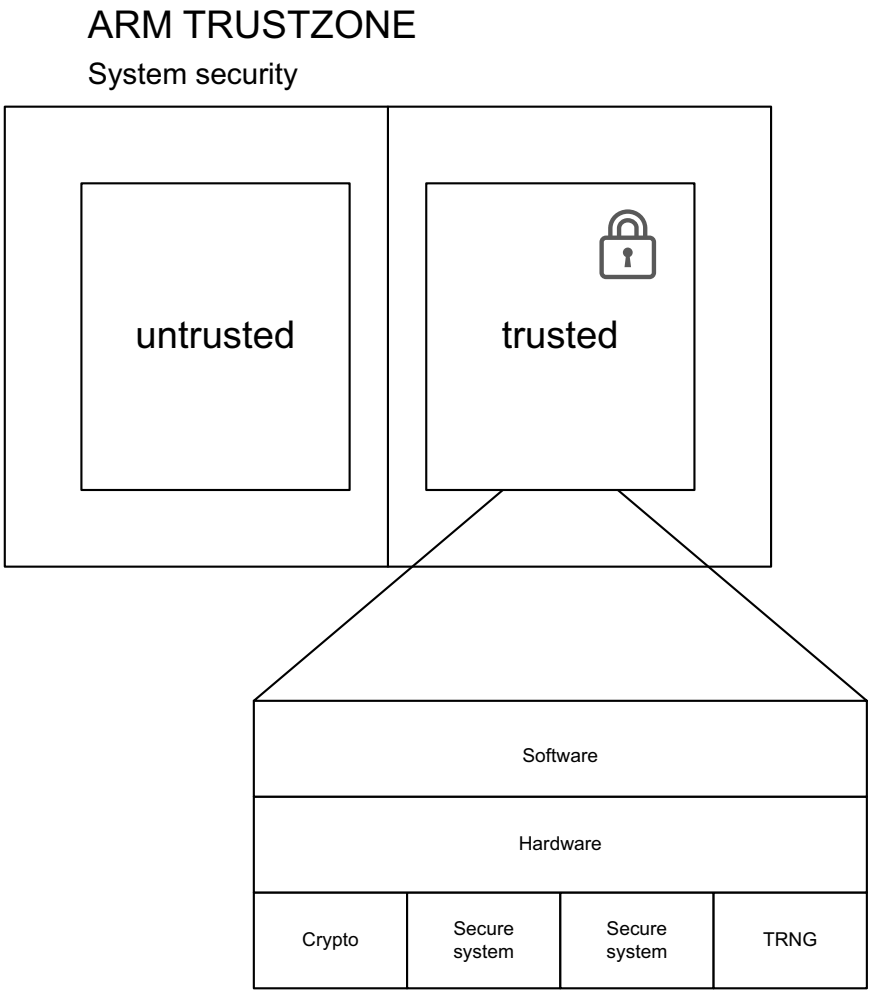


Fig 168. Arm TrustZone system security abstract view

TrustZone for Armv8-M is a new feature on Arm Cortex series that enables execution separation of trusted (Secure) software and access control isolation of trusted resources from non-trusted (Non-secure) software and resources, while running on same CPU. It is achieved by segmentation of memory arrays and peripherals into either Secure (S) or Non-secure (NS). TrustZone for Armv8 is optimized for energy efficient embedded applications that require real-time responsiveness.

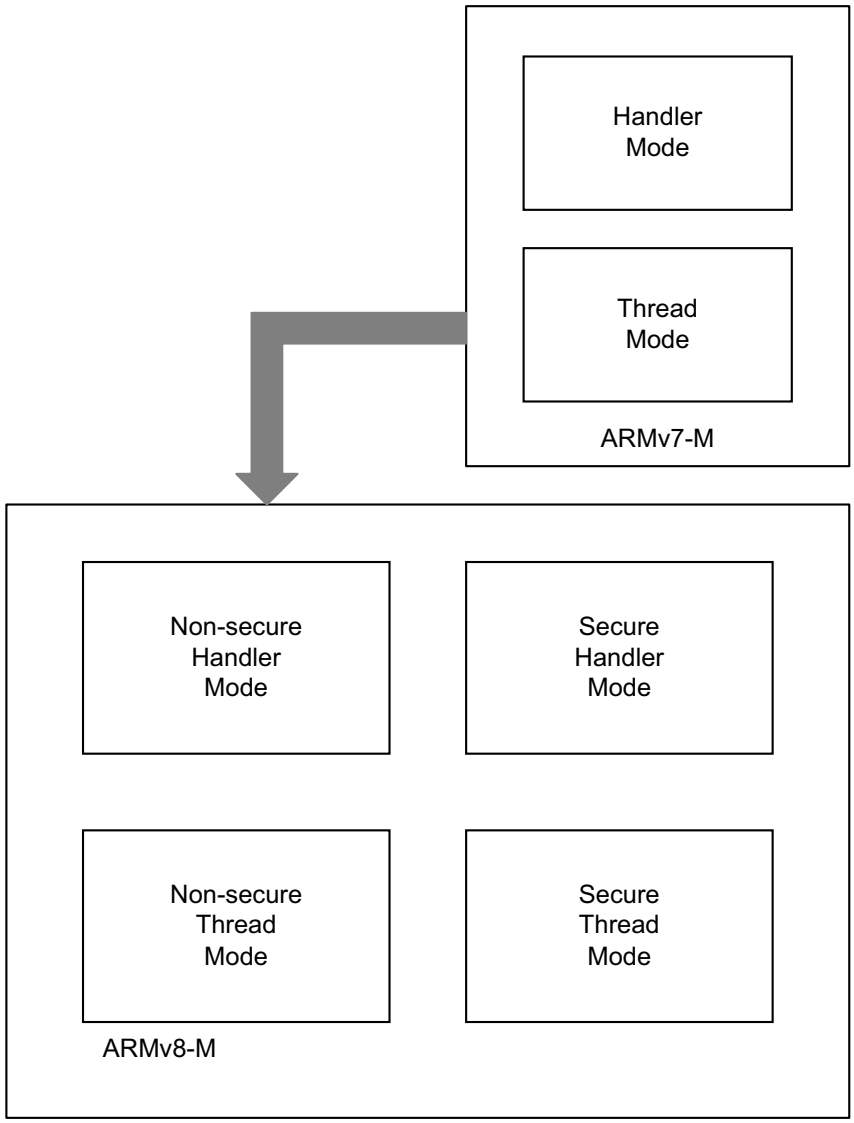


Fig 169. Cortex-M4 (or M3) vs Cortex-M33

More details on TrustZone for ARMv8-M can be found in “TrustZone® technology for ARM®v8-M architecture version 1.0” document at arm.com

The following rules apply to the M33 core if TZ-M functionality is enabled:

- CM33 CPU in Secure state (CPU-S) can execute instructions from secure memory (S-memory) only; it cannot execute from Non-secure memory (NS-memory).
- CPU-S can access data in both S-memory and NS-memory, that is, CPU-S can execute data reads from both S- and NS-memory, as well as execute data writes to S or NS-memory.
- CPU-NS can execute instructions only from NS-memory and cannot execute instructions from S-memory.
- CPU-NS can access data only in NS-memory; that is, CPU-NS can execute data

reads from NS-memory only, and execute data writes to NS-memory only. CPU-NS cannot access data from S-memory.

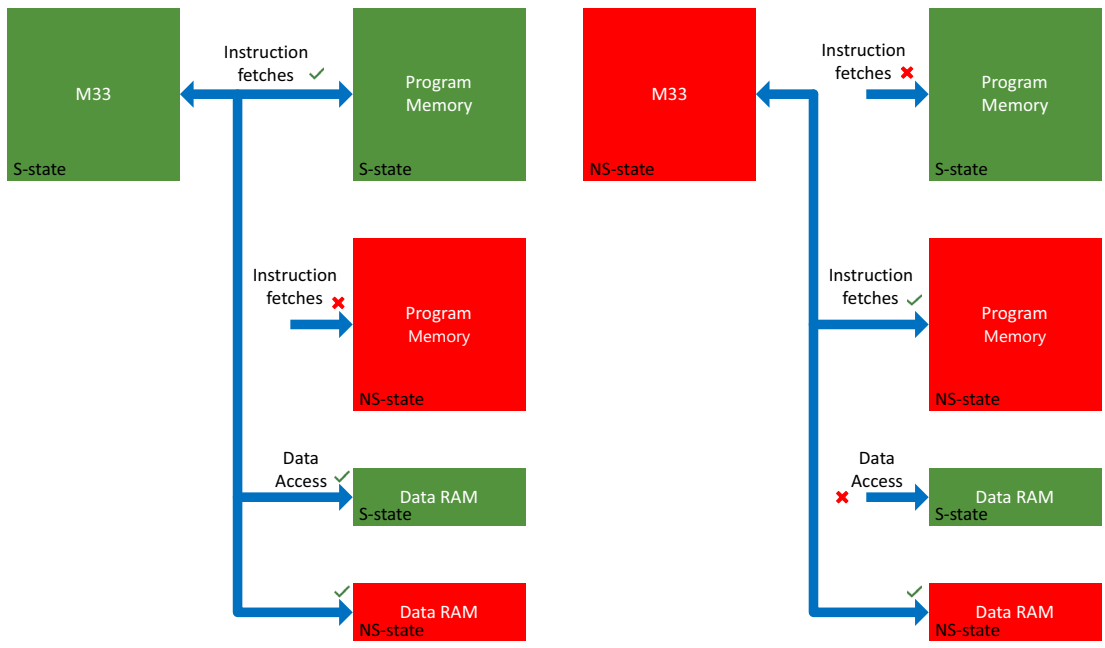


Fig 170. Secure state and Non-secure state view for TrustZone for ARMv8

In summary:

- NS application code *trus*t that secure code will not corrupt or modify NS code or data inadvertently or on purpose to create malfunction or hazard.
- S application code does not *trus*t NS application code and disallows access to a CPU-NS.

To support secure state Cortex-M33 architecture extends to include secure MPU, secure NVIC, secure systick, and secure stack pointer with stack-threshold check.

47.3.1.1 State transitions

At reset release, CPU0 (CM33) is in Secure state.

CPU can call into NS application code from CPU-S state by executing newly introduced instructions:

- BXNS: Branch and Exchange Non Secure – branches to an address in NS-memory.
- BXLNS: Branch with Link Exchange Non Secure - calls a subroutine in NS memory.

On executing either the BXNS or BXLNS instructions the CPU-S will also change to the Non-secure state (CPU-NS) and thus be in the correct state for executing out of NS memory.

CPU cannot access S-memory directly when it is in NS-state. However, TZ-M provides a gateway into S-memory for NS-application code using a special region called Non-Secure Callable (NSC). NSC region lies in S-memory and hence CPU must be in CPU-S state to execute instructions in this region. The NSC region of S-memory provides a veneer for S-application code to access function in S-memory without divulging the specific address of the secure function.

When switching from CPU-NS to CPU-S, an additional gating factor is implemented in form of Secure Gate (SG) instruction. It is placed in NSC region at the start of secure function callable from Non-secure code. The CPU-NS when calling into NSC region must access an address with the SG instruction. The SG instruction is the only instruction that can be executed from a CPU-NS. On executing the SG instruction, the CPU will change from CPU-NS to CPU-S, then will execute the veneered call to a secure function in S-memory. If the CPU-NS calls into an address in the NSC region that is not an SG instruction an exception fault is created. The exception fault results in the CPU entering secure state.

The secure application code developer creates function calls inside the NSC region to S-application code, allowing the NS-application the capability use functions inside S-memory.

47.3.2 Attribution units

TrustZone for ARMv8-M implementation consists of the Security Attribution unit (SAU) and Implementation Defined Attribution Unit (IDAU). Device Attribution Unit (DAU) connects to CPU0 via IDAU interface as shown in Figure 171.

Combination of SAU and IDAU assign a specific security attribute (S, NS, or NSC) to a specific address from the CPU0. Access from CPU0, dependent on its security status and the resultant security attribute set by the IDAU and SAU, is then compared by the secure AHB Controller to a specific checker which marks various access policies for memory and peripherals. secure AHB Controller is discussed in Section 47.3.3.4 “Secure AHB controller”

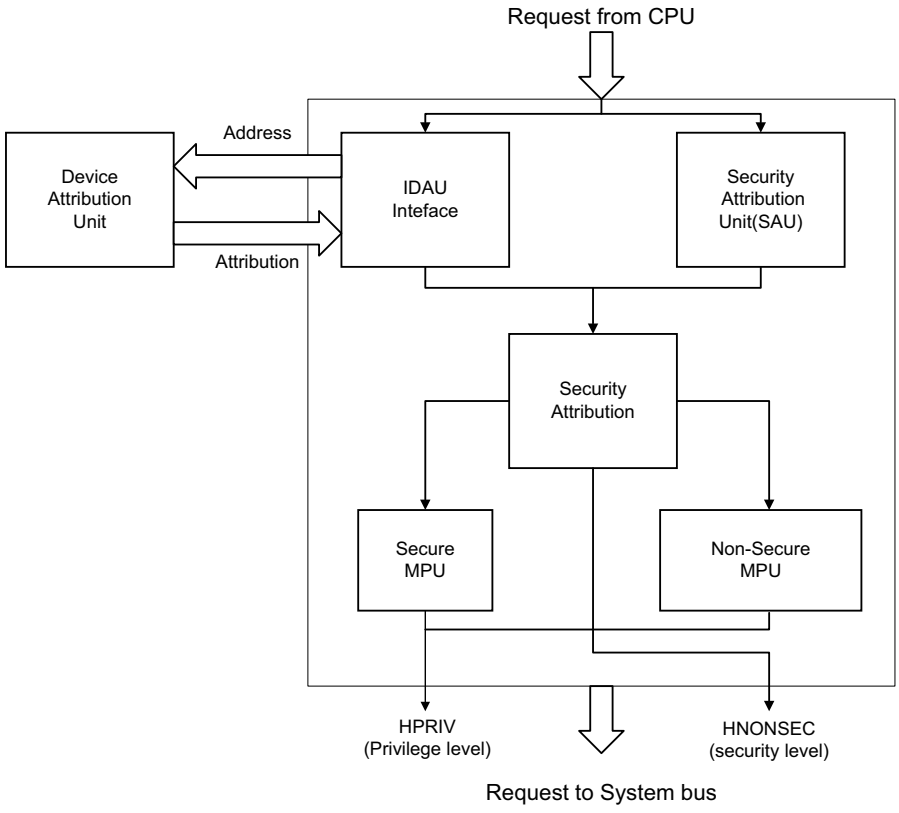


Fig 171. Connection of DAU to CPU0

47.3.2.1 Device Attribution Unit

The LPC55S6x implements a simple Attribution unit that divides the whole memory map into secure or non-secure regions. All peripherals and memories are aliased at two locations.

- Address 0x0000_0000 to 0x1FFF_FFFF
  - Non-secure (always)
- Address 0x2000_0000 to 0xFFFF_FFFF
  - Non-secure if HADDR[28]=0
  - Secure if HADDR[28]=1

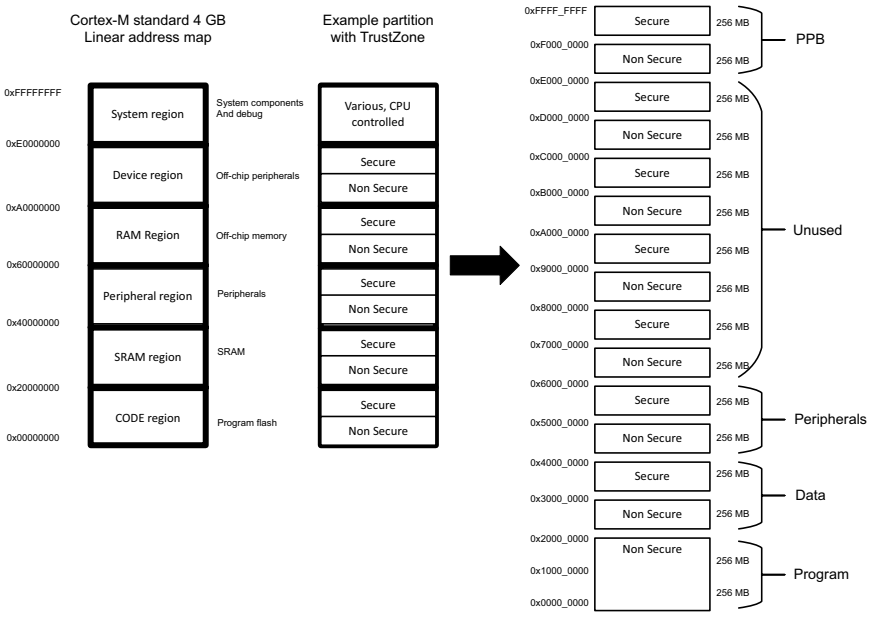


Fig 172. IDAU Implementation

47.3.2.2 Security Attribution Unit

The SAU is internal to CPU0 (CM33 with TZ). It monitors all addresses from the CPU0 and assigns an attribute if this address is S or NS. The SAU does not monitor addresses from bus masters other than the CPU0.

The SAU supports up to eight regional descriptors, each descriptor allows setting security state for a specific memory region from the following attributes.

- S – Secure.
- NS – Non-secure.
- NSC – Non-secure Callable.

However, 0xF000_0000 to 0xFFFF_FFFF range is fixed as secure and SAU cannot program it to be NSC.

The SAU can only be configured by the CPU0 in the secure state. When enabled, the SAU will default all addresses as S. Only secure application code can program descriptor to create NSC or NS regions.

The IDAU works in conjunction with the SAU to assign a specific security attribute (S or NS) to a specific address. Both the IDAU and SAU will respond to a specific address and the CPU0 selects the higher of the two security attributes, where the highest state is Secure and the lowest state is NS. NSC attribute is defined by SAU. In IDAU NSC area can be defined as NS. Regions are aligned to 32-byte boundaries.

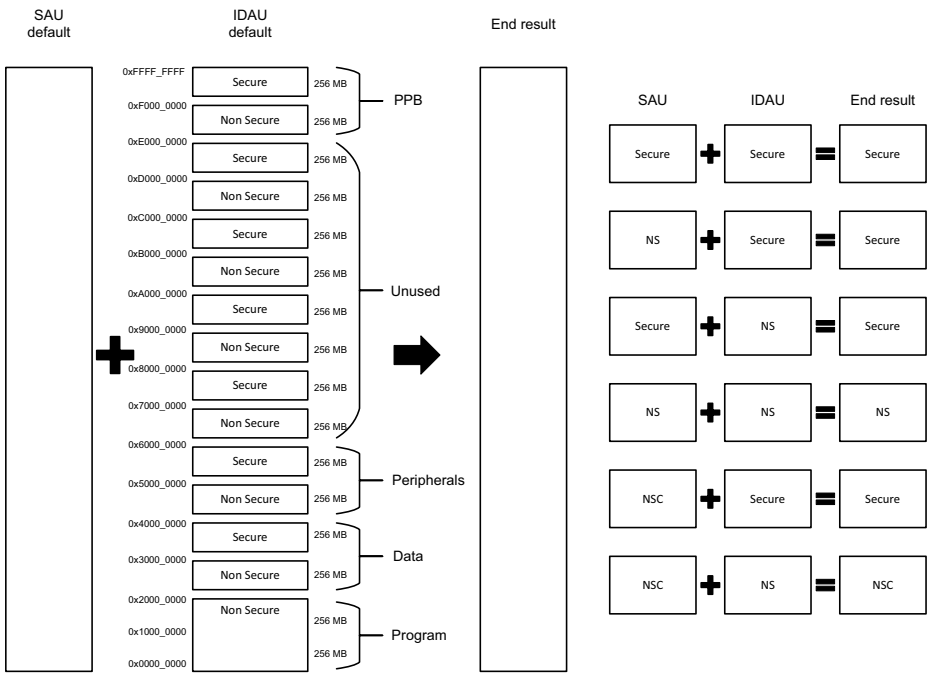


Fig 173. Security attribute definition as combination of SAU and IDAU

From a memory map perspective, the NS address space is an alias of the secure address space for the same physical program memory of 64kB address space at 0x0000_0000 to 0x0000_FFFF, Non-secure application code will fetch instructions in the 0x0000_0000 to 0x0000_FFFF Non-secure (NS) space (address bit28 = 0) if the physical address space is configured as Non-secure, where secure application code will execute in 0x1000_0000 to 0x1000_FFFF Secure (S) space if the physical address space is configured as secure. Similarly, secure application code will access all peripherals in the 0x5000_0000 to 0x5FFF_FFFF space (address bit28=1), and NS application code will access NS peripherals at 0x4000_0000 to 0x4FFF_FFFF space. Details of SAU programmable registers can be found in ARM CM33 documents.

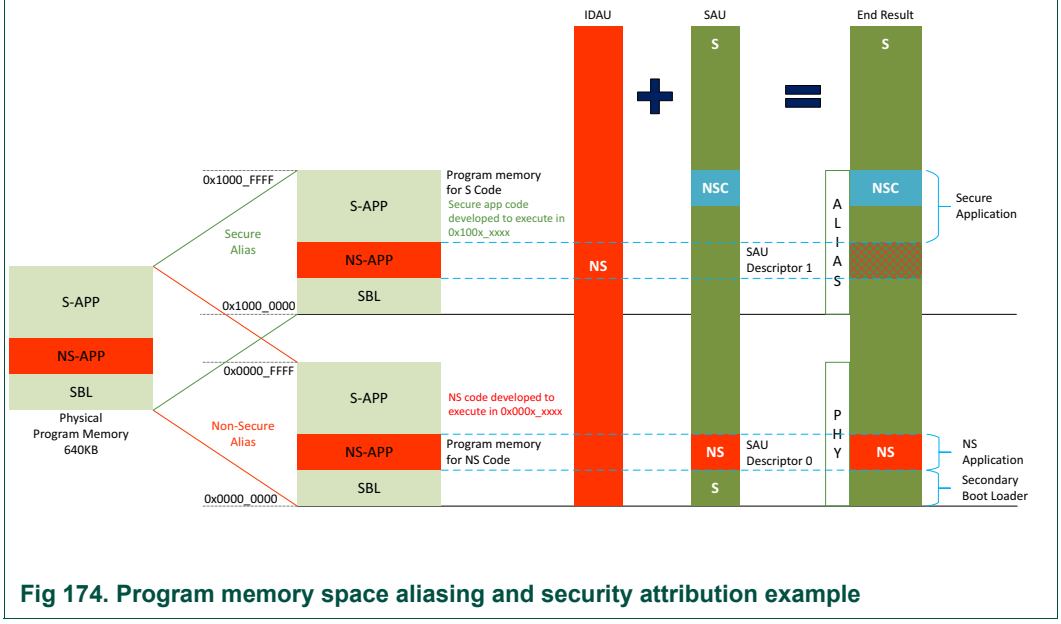


Fig 174. Program memory space aliasing and security attribution example

47.3.2.3 Region number and test target instruction

The IDAU generates a Region Number (RN) for each region, which can be used by application code to determine security level of that region. RN is a 8-bit number. In LPC55xxx IDAU returns region number as:

IDAU_RN[7:0] = ({4'h0, idau_addr_a[31:28]})

SAU will also return information on TT instruction indicating NS or S attribute.

Application can use Test Target instruction (TT) on start and end address of a region. Instruction returns RN and security attributes (NS or S).

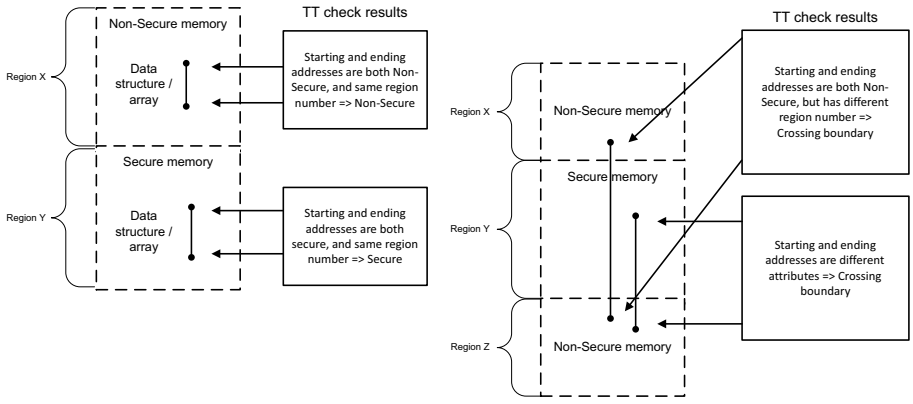


Fig 175. Test Target instruction usage

47.3.3 Secure AHB bus and secure AHB Controller

CM33 TZ-M implementation consists of the IDAU and SAU modules, which filters address access from CPU0 based on specific security attribute (S, NS, or NSC) assigned to that address space. The LPC55S6x implements second layer of protection with secure AHB Bus to support secure trusted execution at system-level.

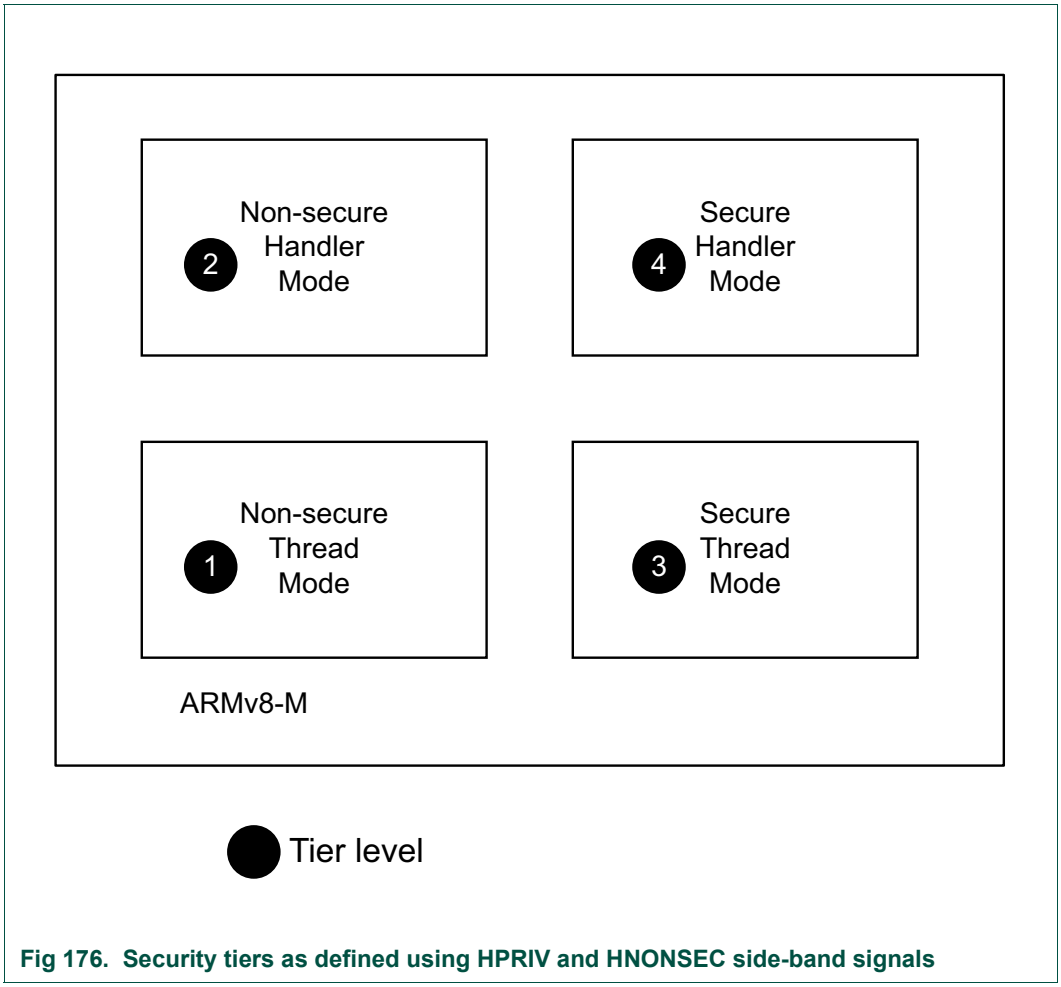


Fig 176. Security tiers as defined using HPRIV and HNONSEC side-band signals

The secure AHB Controller provides access policies for all the bus slaves via checker functions. All masters on LPC55S6x outputs security side-band signals HPRIV (privileged) and HNONSEC (Secure access) as indication of security attributes for a given access. Secure AHB Bus processes this signals and compares them against security attributes set for slaves in secure AHB Controller. Access is granted if security attribute of requested access is not violating the security attribute of the slave being accessed. Security violation interrupt is raised if violation occurs on data or instruction access. CPU0 switches to secure mode to handle the violation.

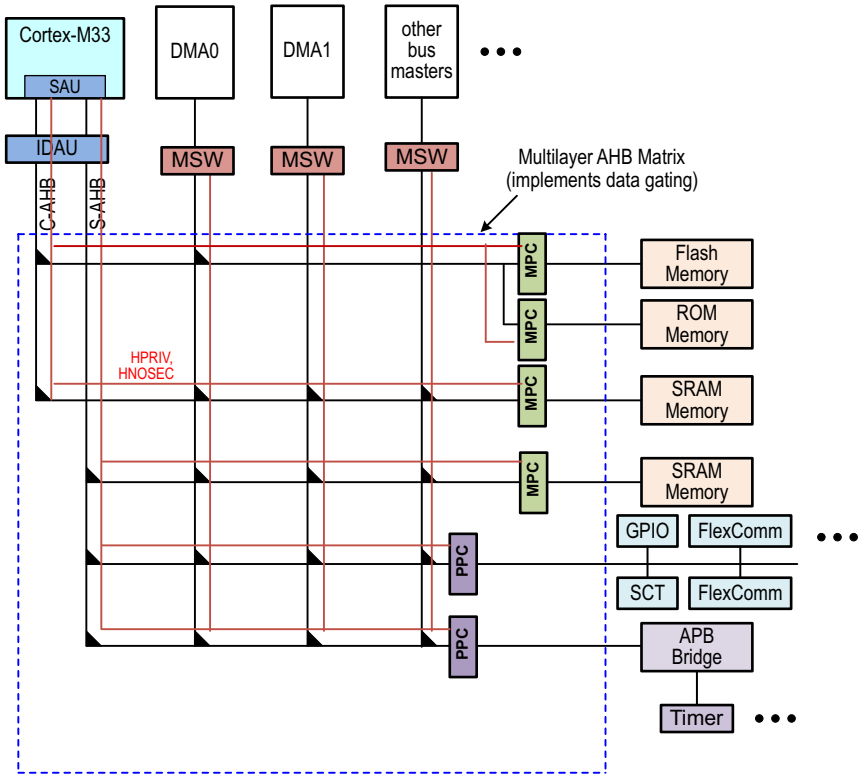


Fig 177. System view with secure AHB bus

These side-band signals create four security tiers as depicted in Figure 177. Data accesses are allowed from higher tier master to same or lower tier slave. However, instruction accesses are stricter, a master can access a slave only at same security tier. A special programmable option is available that allows treating all access in the system as instruction, meaning data access checker can also be as strict.

This protection is achieved using three primary components:

47.3.3.1 Memory Protection Checkers (MPC)

On the LPC55S6x on-chip flash, ROM and RAM can be protected against access from the application with lower tier security using Memory Protection Checkers (MPC). ROM and each RAM bank has associated MPC. Flash has one MPC. Flash, ROM and each RAM bank are divided into smaller sub-region to offer more granularity for security tier assignment and filtering. ROM as well as each RAM is divided into 4 kB sub-regions and Flash is divided into 32 kB sub-regions. Each sub-region can be assigned individual security tier by programing corresponding registers in secure AHB controller.

Table 883. Security tier granularity for on-chip memories

Memory	Total Size	Sub-region size	Sub-regions count
Flash	640 kB	32 kB	20
ROM	128 kB	4 kB	32
SRAM-X	32 kB	4 kB	8
SRAM-0	64 kB	4 kB	16

Table 883. Security tier granularity for on-chip memories *…continued*

Memory	Total Size	Sub-region size	Sub-regions count
SRAM-1	64 kB	4 kB	16
SRAM-2	64 kB	4 kB	16
SRAM-3	64 kB	4 kB	16
SRAM-4	16 kB	4 kB	4

47.3.3.2 Peripheral Protection Checkers (PPC)

On the LPC55S6x, all peripherals on the AHB slave port, (as well as each peripheral on the APB bridge), can be protected against access from the application with lower tier security using Peripheral Protection Checkers (PPC). Each AHB port has associated PPC that offers granularity at individual slave levels for security tier assignment and filtering. Each APB bridge has an associated PPC that offers granularity for security tier assignment and filtering for each APB peripheral. Each peripheral can be assigned individual security tier by programing corresponding registers in secure AHB Controller.

47.3.3.3 Master Security Wrapper (MSW)

TrustZone for ARMv8-M offers IDAU functionality for CM33 when TrustZone feature is configured. However, IDAU is not available for all other masters on LPC55S6x. A special wrapper Master Security Wrapper (MSW) is implemented for each AHB Master except CPU0.

MSW allows application to set security attribute for each master. There are two categories of the MSW:

1. Simple Master: Bus Masters that can perform data access only: SDIO, USB-FS, DMA0, DMA1, Hash-AES.

MSW for simple master enables strict checking by default. Secure data accesses can access secure memory only. A programmable option to disable strict checking allows data access from secure master to Non-secure memory.

Security level as defined in MSW are supposed to be static, it is programmed by application once and locked until next system reset. Hash-AES is one exception, this master allows dynamic re-programing of security tier to allow the functionality to be used by secure and non-secure code. Special measure is in place within the module to guard against malicious intend, only a master with higher or same security tier update security attributes. New attribute cannot be higher than that of programing master. Buffers within the module are flushed before switching security attributes.

If a master is programmed to be secure master, it must output the address with AHB address bit 28 to be 1.

47.3.3.4 Secure AHB controller

Secure AHB controller is a module on LPC55S6x at memory offset 0x400A-C000. It allows programming security attributes for all MPCs, PPCs in addition to MSWs.

It also supports locking of SAU setting, secure and Non-secure MPU settings (MPU_S/MPU_NS), secure and Non-secure vector offset settings (VTOR_S/VTOR_NS) for CPU0 and MPU_NS and VTOR_NS setting for CPU1. This enables BootROM to safeguard certain security features and disable possibility of enabling those dynamically by unintentionally or with malicious intent.

It also supports register programing for GPIO masking and Interrupt masking. Details are described in Section 47.4 “Register description”.

When a security violation is detected interrupt is raised by secure AHB Controller module. It also logs violation information such as address being accessed on a AHB slave port when violation occurred as well as access type and security attributes of the master that generated unauthorized access.

Only application that can write to secure AHB Controller to configure security attributes is the one that has secure and privileged access rights. Hence, from application perspective, highest tier (tier-4) thread from CPU0 can program security attributes for system slaves.

Registers programmed in secure AHB Controller are retained during deep-sleep and power-down modes, however these registers need re-programing after wake-up from deep-power down.

47.3.4 Interrupt, DMA and GPIO: Secure instance and masking

The LPC55S6x has two CPUs. CPU0 is with TrustZone and CPU1 is without TrustZone. By default, both CPUs have access to all interrupts. If CPU1 is configured as non-secure master using MSW, NS code can have access to interrupt generated by secure peripheral. To safeguard secure application, interrupt masking feature is implemented on the LPC55S6x. Any interrupt to CPU1 can be masked out by programming SEC_CPU_INT_MASK0/1 registers at offset 0xF90 and 0xF94 in secure AHB Controller module. CPU0 is with TZ and hence has secure NVIC (NVIC_NS) and non-secure NVIC (NVIC_NS). CPU0 has internal programmability to configure any interrupt as secure interrupt, making it visible to NVIC_S only and mask it from NVIC_NS. Refer ARM CM33 documents for more details.

The LPC55S6x has two DMA instances. DMA0 offers 23 channels and DMA1 offers 10 channels. Either of the DMA can be selected as secure DMA, selection depends on DMA needs of relevant secure peripherals selected. To disable DMA Request from secure peripherals to be visible to non-secure DMA, DMA masking feature is implemented. DMA masking can be programmed using registers in Chapter 18 “LPC55S6x/LPC55S2x/LPC552x Input Multiplexing (INPUTMUX)”or Chapter 22 “LPC55S6x/LPC55S2x/LPC552x DMA controller”

On the LPC55S6x, all digital IO pins states are readable through GPIO-HS module, independent of which function is chosen using I/O multiplexer. It can lead to information leak in case a secure peripheral is connected to the interface. To safeguard incoming data on secure peripherals, GPIO masking is implemented on the LPC55S6x. Any digital I/O that is sensitive to information leakage can be masked using SEC_GPIO_MASK0/1/2/3 registers at offset 0xF80-0xF8C in secure AHB Controller module.

Additionally, LPC55S6x also has additional instance of GPIO-HS and GPIO-PINT module on Port0 (0-31). Unlike normal GPIO, this GPIO functionality is implemented as IOMUX function and available only if selected using IOCON programing. It can be used as secure GPIO to generate certain input pattern from external device for secure signaling. More details can be found in Chapter 16 “LPC55S6x/LPC55S2x/LPC552x General Purpose I/O (GPIO)” and Chapter 20 “LPC55S6x/LPC55S2x/LPC552x Secure pin interrupt and pattern match (Secure PINT)” chapters.

47.3.5 TrustZone configuration example

The LPC55S1x/LPC551x implements two layers of TrustZone protection. The first layer consists of two attribution units (IDAU and SAU) on the CM33 core. The second layer consist of a secure AHB bus and AHB secure controller implemented at the system level. The proper configuration of both layers is essential for secure trusted environment execution. The TrustZone configuration is illustrated using a simplified MCU as shown in Figure 178 “Example of cortex-M33 device with security extension”, but the same principles can be applied. The simplified device consists of a CM33 core with the security extension running in secure or normal mode, 64KB of code memory, 64KB data memory and four peripherals. The peripherals, code, and data memory are connected to the core via a secure AHB bus. Due to memory address aliasing (using address bit A28), the CM33 core sees all resources (all memories and peripherals) twice in the memory map. For example, the 64KB code memory can be seen either as 0x0000 - 0xFFFF or 0x1000_0000 - 0x1000_FFFF. To keep figures simple, the data memory (0x2000_0000 - 0x2000_FFFF, 0x3000_0000 - 0x3000_FFFF) is not shown on all figures.

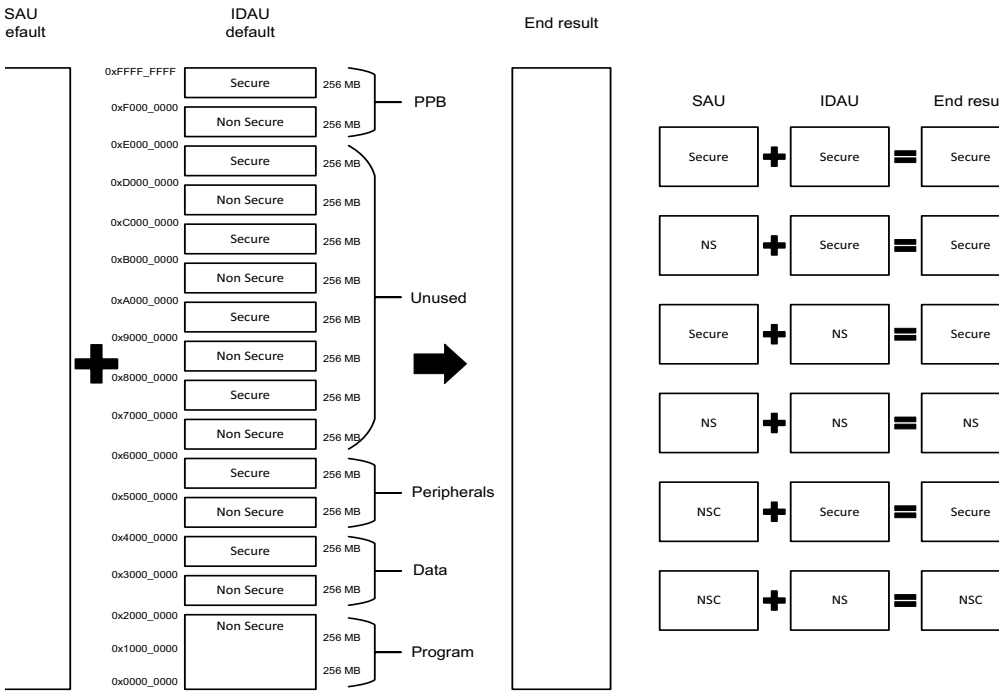


Fig 178. Example of cortex-M33 device with security extension

Before performing a TrustZone configuration, you must define which MCU resources (memories and peripherals) are to be relegated as secure and which ones as non-secure depending on the particular environment. For this example, the trusted execution environment is defined as:

- The first 32KB of code memory (0x0000 - 0x7FFF) as non-secure memory.
- The second 32KB of code memory (0x8000 - 0xFFFF) as secure memory.
- The 48KB of data memory (0x2000_0000 - 0x2000_BFFF) as non-secure.

- The 16KB of data memory (0x2000_C000 - 0x2000_FFFF) as secure memory
- SPI and TIMER as secure peripherals.
- UART and ADC as non-secure peripherals.

The rest of the address space remains secure.

After the appropriate TrustZone configuration, the MCU resources will be available for the addresses shown in Table 884 “MCU memory layout after TrustZone configuration”:

Table 884. MCU memory layout after TrustZone configuration

MCU resource	Address range	Security attribute
Non-secure code memory	0x0000_0000 - 0x0000_7FFF	Non-secure
Secure code memory	0x1000_8000 - 0x1000_FFFF	Secure
Non-secure data memory	0x2000_0000 - 0x2000_BFFF	Non-secure
Secure data memory	0x3000_C000 - 0x3000_FFFF	Secure
UART	0x4000_0000 - 0x4000_0FFF	Non-secure
TIMER	0x5000_1000 - 0x5000_1FFF	Secure
SPI	0x5000_2000 - 0x5000_2FFF	Secure
ADC	0x4000_3000 - 0x4000_3FFF	Non-secure

The TrustZone configuration starts with a secure attribution map definition, which splits MCU resources (MCU memories and peripherals) between secure and non-secure domains. The security attribution map is defined by IDAU and SAU. The default security attribution map is defined by IDAU and it can be modified by the SAU configuration. If the SAU is disabled or empty (none of SAU region is configured and enabled), the whole address space is secure. To assign some address spaces into a non-secure domain, this address space must be configured in the SAU and the same address space must be marked as no-secure in IDAU. The address space marked as secure in IDAU can never be assigned to non-secure domain.

The simplest way to set up SAU is to configure every non-secure contiguous address space as one region in SAU. Using this approach, the SAU will be configured as shown in Table 885 “Basic SAU configuration”.

Table 885. Basic SAU configuration

	RBAR	RLAR		
	Base Address	Limit Address	NSC	ENABLE
Region 0 (code memory)	0x0000_0000	0x0000_7FFF	0	1
Region 1 (NSC memory)	0x1000_FC00	0x1000_FFFF	1	1
Region 2 (data memory)	0x2000_0000	0x2000_BFFF	0	1
Region 3 (UART)	0x4000_0000	0x4000_0FFF	0	0
Region 4 (TIMER)	0x4000_3000	0x4000_3FFF	0	0

The security attribution map after SAU configuration can be seen on Figure 179 “TrustZone isolation after basic SAU configuration”. The SAU configuration also includes 1KB of secure, non-secure callable (NSC) memory placed on the top of secure memory. This region is used for a veneer table. The veneer table contains all secure functions/services, which are callable from non-secure environment. For the purposes of this explanation, the NSC region is not shown on all figures or for data memory.

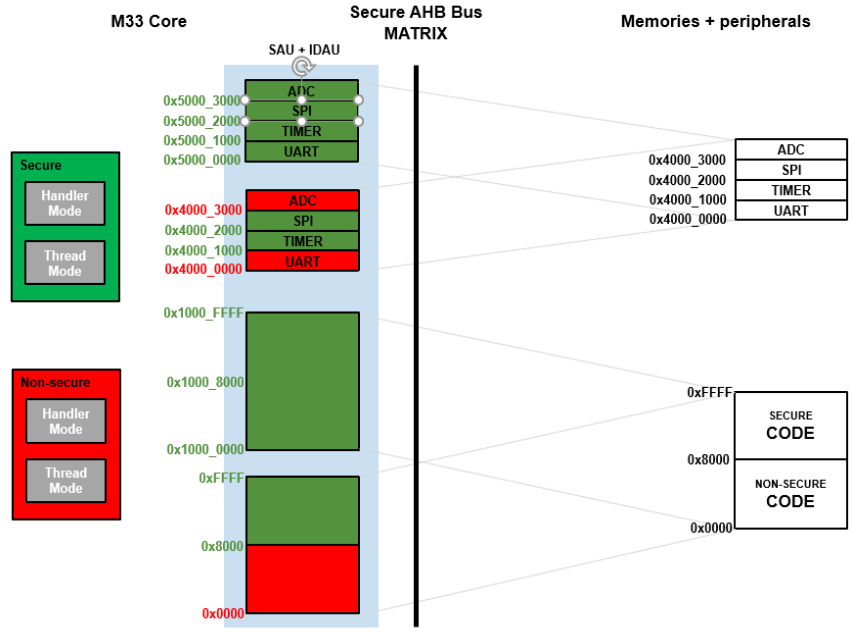


Fig 179. TrustZone isolation after basic SAU configuration

After basic SAU configuration, the TrustZone isolation is fully functional with the exception of the following limitations:

- Every contiguous memory space requires one SAU region. Since there are only 8 SAU regions, this approach is suitable for simple applications.
- Since IDAU/SAU only manages CM33 core transactions, there is no information regarding TrustZone isolation at the system level. As a result, TrustZone isolation is unknown to other bus masters in the system such as DMA, USB, etc.
- TrustZone isolation only relies on SAU configuration. In case of an invalid SAU configuration (due to a software error or glitch attack) TrustZone isolation is also corrupted.

Due to these limitations, this way of trusted execution environment configuration (based on basic SAU set up only) is not highly recommended and second layer protection using secure AHB controller must be employed. Second protection layer brings much higher flexibility in TrustZone configuration and higher isolation security as will be shown in the rest of this chapter.

Keep in mind that real world applications are much more complex than what is presented in this example. Peripheral configuration can be especially challenging given that there are only 8 SAU regions. Therefore, a different approach for SAU configuration must be chosen. This approach is also called canonical SAU configuration. It relies on the principle that all MCU resources (all memories and peripherals) have one secure and one non-secure alias. In other words, every region in standard Cortex-M address space is

divided into halves, where address space with address bit A28=1 denotes a secure address while A28=0 indicates a non-secure address. A canonical SAU configuration is shown in Table 886 “Canonical SAU configuration”.

Table 886. Canonical SAU configuration

	RBAR	RLAR		
	Base Address	**Limit Address**	**NSC**	**ENABLE**
Region 0 (code memory)	0x0000_0000	0x1FFF_FFFF	0	1
Region 1 (data memory)	0x2000_0000	0x5FFF_FFFF	0	1

A canonical SAU configuration only requires two regions so there are still six SAU regions available for other purposes, for example for NSC regions definition. An example is shown in Figure 180 “TrustZone isolation after canonical SAU configuration”, configured using canonical SAU configuration..

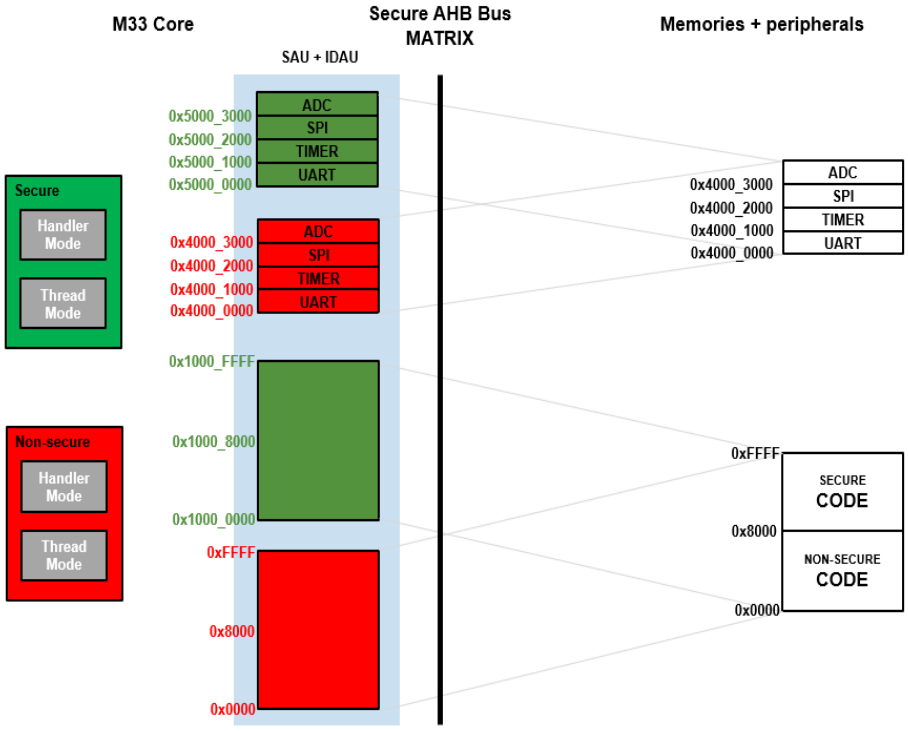


Fig 180. TrustZone isolation after canonical SAU configuration

Comparing Figure 179 “TrustZone isolation after basic SAU configuration” and Figure 180 “TrustZone isolation after canonical SAU configuration” it can be seen, that TrustZone isolation is not functional now. For example, secure code in address range 0x1000_8000 - 0x1000_FFFF is also available in non-secure mode in address range 0x8000 - 0xFFFF. This is because the SAU configuration is visible on a core level only and thus it is not known whether address range 0x8000 - 0xFFFF is assigned to a secure or non-secure domain. Moreover, the address range 0x8000 - 0xFFFF is assigned to both domains on the core level. To make TrustZone isolation functional, the AHB secure controller must be configured and enabled.

The secure AHB controller implements trusted execution protection on a system level. The memory and peripheral protection checkers assign MCU resources either to secure or non-secure domains. Both checkers work in the same way with the difference being that memory protection checker (MPC) divides memory into sub-regions while memory peripheral checker (PPC) divides memory per individual peripherals. Every memory sub-region or peripheral has its own security access rule, which assigns it to the specific security domain/level.

- "Secure - privilege
- "Secure - non-privilege
- "Non-secure - privilege
- "Non-secure - non-privilege

The privilege/non-privilege check is optional and allows to define four tier levels.

The AHB secure controller configuration for this example is shown in Figure 181 “TrustZone isolation after canonical form of SAU and secure AHB controller configuration”.

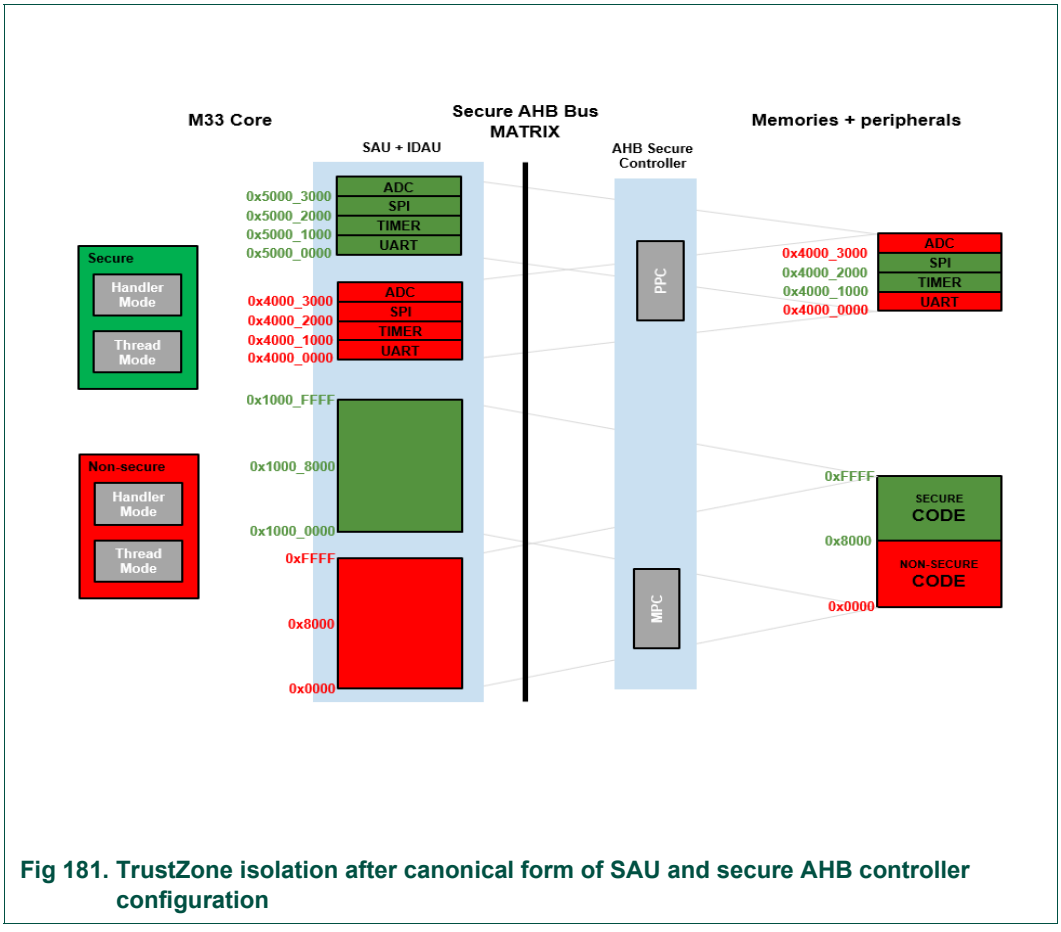


Fig 181. TrustZone isolation after canonical form of SAU and secure AHB controller configuration

After proper AHB secure controller configuration, the TrustZone isolation is fully functional. Compared with the configuration shown in Figure 180 “TrustZone isolation after canonical SAU configuration”, the non-secure software can still complete a

non-secure transaction with address 0x8000, but this transaction is blocked by the AHB controller because the memory with physical address 0x8000 is already assigned to secure domain. This means that the secure code is accessible via the secure alias (0x1000_8000 - 0x1000FFFF) only. Another effect of memory and peripheral checkers is also a higher configuration flexibility. With a canonical SAU configuration, you are not limited by the number of SAU regions and flexibility is enhanced up to a memory sub-region or peripheral level. Besides protection checkers, the AHB secure controller implements a simplified IDAU for all non-core bus master (MSW) functions so they can also benefit from TrustZone isolation.

The last SAU configuration example combines both previous approaches together with an AHB secure controller. The advantage of basic SAU configuration together with enabled AHB secure controller is a cross checking of the bus transaction. The first check is done at the core level (SAU/IDAU) while the second check is performed at the system level (AHB secure controller). If some inconsistency is detected between the SAU and AHB secure controller configurations (due to some software error or glitch attack), access to specific resource is blocked. So, by employing both SAU and AHB secure controller for TrustZone isolation makes isolation more robust when compared to a basic SAU configuration shown in Figure 179 “TrustZone isolation after basic SAU configuration”.

The TrustZone example shown in Figure 182 “TrustZone isolation after combined SAU and secure AHB controller configuration” uses basic SAU configuration for code and data memories, and canonical SAU configuration for peripherals. This combination provides the highest isolation robustness and keeps sufficient flexibility even for complex applications.

Table 887. Combined SAU configuration

	RBAR	RLAR		
	Base Address	**Limit Address**	**NSC**	**ENABLE**
Region 0 (code memory)	0x0000_0000	0x0000_7FFF	0	1
Region 1 (NSC memory)	0x1000_FC00	0x1000_FFFF	1	1
Region 2 (data memory)	0x2000_0000	0x2000_BFFF	0	1
Region 3 (peripheral memory)	0x4000_0000	0x5FFF_FFFF	0	1

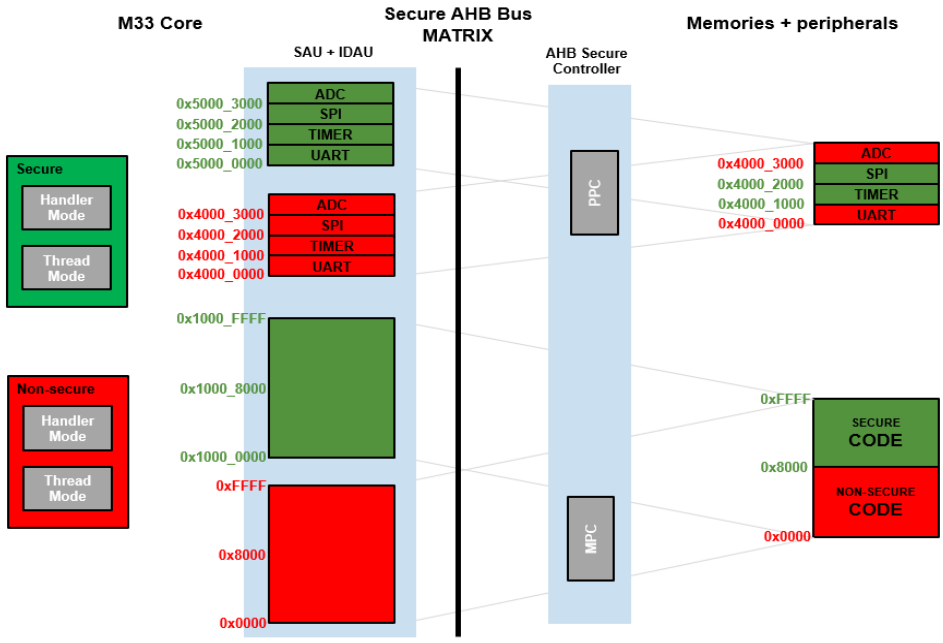


Fig 182. TrustZone isolation after combined SAU and secure AHB controller configuration

47.4 Register description

Table 888. Register overview: AHB_Secure_CTRL (base address = 0x500AC000) [1] [2]

Name	Access	Offset	Description	Reset value	Section
SEC_CTRL_FLASH_ROM_SLAVE_RULE	RW	0x0	Security access rules for the slave port that connects FLASH and ROM memories. This rule supersede more granular security rules as in SEC_CTRL_FLASH_MEM_RULEx	0x0	47.4.1
SEC_CTRL_FLASH_MEM_RULE0	RW	0x10	Security access rules for FLASH sector 0 to sector 7, address range 0x0000_0000 - 0x0003_FFFF. Each Flash sector is 32 kbytes. There are 20 FLASH sectors in total.	0x0	47.4.2
SEC_CTRL_FLASH_MEM_RULE1	RW	0x14	Security access rules for FLASH sector 8 to sector 15, address range 0x0004_0000 - 0x0007_FFFF. Each Flash sector is 32 kbytes. There are 20 FLASH sectors in total	0x0	47.4.3

Table 888. Register overview: AHB_Secure_CTRL (base address = 0x500AC000) *...continued* [1] [2]

Name	Access	Offset	Description	Reset value	Section
SEC_CTRL_FLASH_MEM_RULE2	RW	0x18	Security access rules for FLASH sector 16 to sector 19, address range 0x0008_0000 - 0x0009_FFFF. Each Flash sector is 32 kbytes. There are 20 FLASH sectors in total.	0x0	47.4.4
SEC_CTRL_ROM_MEM_RULE0	RW	0x20	Security access rules for ROM sector 0 to sector 7, address range 0x0300_0000 - 0x0300_7FFF. Each ROM sector is 4 kbytes. There are 32 ROM sectors in total.	0x0	47.4.5
SEC_CTRL_ROM_MEM_RULE1	RW	0x24	Security access rules for ROM sector 8 to sector 15, address range 0x0300_8000 - 0x0300_FFFF. Each ROM sector is 4 kbytes. There are 32 ROM sectors in total	0x0	47.4.6
SEC_CTRL_ROM_MEM_RULE2	RW	0x28	Security access rules for ROM sector 16 to sector 23, address range 0x0301_0000 - 0x0301_7FFF.	0x0	47.4.7
SEC_CTRL_ROM_MEM_RULE3	RW	0x2C	Security access rules for ROM sector 24 to sector 31, address range 0x0301_8000 - 0x0302_FFFF. Each ROM sector is 4 kbytes. There are 32 ROM sectors in total.	0x0	47.4.8
SEC_CTRL_RAMX_SLAVE_RULE	RW	0x30	Security access rules for RAMX slave. This rule supersede more granular security rules as in SEC_CTRL_RAMX_MEM_RULEx	0x0	47.4.9
SEC_CTRL_RAMX_MEM_RULE0	RW	0x40	Security access rules for sub-regions within RAMX.	0x0	47.4.10
SEC_CTRL_RAM0_SLAVE_RULE	RW	0x50	Security access rules for RAM0 slave. This rule supersede more granular security rules as in SEC_CTRL_RAM0_MEM_RULEx	0x0	47.4.11
SEC_CTRL_RAM0_MEM_RULE0	RW	0x60	Security access rules for sub-regions within RAM0.	0x0	47.4.12
SEC_CTRL_RAM0_MEM_RULE1	RW	0x64	Security access rules for sub-regions within RAM0	0x0	47.4.13

Table 888. Register overview: AHB_Secure_CTRL (base address = 0x500AC000) *...continued* [1] [2]

Name	Access	Offset	Description	Reset value	Section
SEC_CTRL_RAM1_SLAVE_RULE	RW	0x70	Security access rules for RAM1 slave. This rule supersede more granular security rules as in SEC_CTRL_RAM1_MEM_RULEx	0x0	47.4.14
SEC_CTRL_RAM1_MEM_RULE0	RW	0x80	Security access rules for sub-regions within RAM1.	0x0	47.4.15
SEC_CTRL_RAM1_MEM_RULE1	RW	0x84	Security access rules for sub-regions within RAM1.	0x0	47.4.16
SEC_CTRL_RAM2_SLAVE_RULE	RW	0x90	Security access rules for RAM2 slave. This rule supersede more granular security rules as in SEC_CTRL_RAM2_MEM_RULEx	0x0	47.4.17
SEC_CTRL_RAM2_MEM_RULE0	RW	0xA0	Security access rules for sub-regions within RAM2.	0x0	47.4.18
SEC_CTRL_RAM2_MEM_RULE1	RW	0xA4	Security access rules for sub-regions within RAM2.	0x0	47.4.19
SEC_CTRL_RAM3_SLAVE_RULE	RW	0xB0	Security access rules for RAM3 slave. This rule supersede more granular security rules as in SEC_CTRL_RAM3_MEM_RULEx.	0x0	47.4.20
SEC_CTRL_RAM3_MEM_RULE0	RW	0xC0	Security access rules for sub-regions within RAM3.	0x0	47.4.21
SEC_CTRL_RAM3_MEM_RULE1	RW	0xC4	Security access rules for sub-regions within RAM3.	0x0	47.4.22
SEC_CTRL_RAM4_SLAVE_RULE	RW	0xD0	Security access rules for RAM4 slave. This rule supersede more granular security rules as in SEC_CTRL_RAM4_MEM_RULEx	0x0	47.4.23
SEC_CTRL_RAM4_MEM_RULE0	RW	0xE0	Security access rules for sub-regions within RAM4.	0x0	47.4.24
SEC_CTRL_APB_BRIDGE_SLAVE_RULE	RW	0xF0	Security access rules for APB Bridge slave. This rule supersede more granular security rules as in SEC_CTRL_APB_BRIDGEx_MEM_CTRLx	0x0	47.4.25
SEC_CTRL_APB_BRIDGE0_MEM_CTRL0	RW	0x100	Security access rules for individual peripherals on APB Bridge 0. Each peripheral can be assigned independent security level.	0x0	47.4.26

Table 888. Register overview: AHB_Secure_CTRL (base address = 0x500AC000) *...continued* [1] [2]

Name	Access	Offset	Description	Reset value	Section
SEC_CTRL_APB_BRIDGE0_MEM_CTRL1	RW	0x104	Security access rules for individual peripherals on APB Bridge 0. Each peripheral can be assigned independent security level.	0x0	47.4.27
SEC_CTRL_APB_BRIDGE0_MEM_CTRL2	RW	0x108	Security access rules for individual peripherals on APB Bridge 0. Each peripheral can be assigned independent security level.	0x0	47.4.28
SEC_CTRL_APB_BRIDGE1_MEM_CTRL0	RW	0x110	Security access rules for individual peripherals on APB Bridge 1. Each peripheral can be assigned independent security level.	0x0	47.4.30
SEC_CTRL_APB_BRIDGE1_MEM_CTRL1	RW	0x114	Security access rules for individual peripherals on APB Bridge 1. Each peripheral can be assigned independent security level.	0x0	47.4.31
SEC_CTRL_APB_BRIDGE1_MEM_CTRL2	RW	0x118	Security access rules for individual peripherals on APB Bridge 1. Each peripheral can be assigned independent security level..	0x0	47.4.32
SEC_CTRL_APB_BRIDGE1_MEM_CTRL3	RW	0x11C	Security access rules for individual peripherals on APB Bridge 1. Each peripheral can be assigned independent security level.	0x0	47.4.33
SEC_CTRL_AHB_LAYER8_SLAVE0_RULE	RW	0x120	Security access rules for AHB peripherals. Each peripheral can be assigned independent security level	0x0	47.4.34
SEC_CTRL_AHB_LAYER8_SLAVE1_RULE	RW	0x124	Security access rules for AHB peripherals. Each peripheral can be assigned independent security level	0x0	47.4.35
SEC_CTRL_AHB_LAYER9_SLAVE0_RULE	RW	0x130	Security access rules for AHB peripherals. Each peripheral can be assigned independent security level	0x0	47.4.36
SEC_CTRL_AHB_LAYER9_SLAVE1_RULE	RW	0x134	Security access rules for AHB peripherals. Each peripheral can be assigned independent security level	0x0	47.4.37
SEC_CTRL_AHB_LAYER10_SLAVE0_RULE	RW	0x140	Security access rules for AHB peripherals. Each peripheral can be assigned independent security level	0x0	47.4.38
SEC_CTRL_AHB_LAYER10_SLAVE1_RULE	RW	0x144	Security access rules for AHB peripherals. Each peripheral can be assigned independent security level	0x0	47.4.39
SEC_CTRL_AHB_SEC_CTRL_MEM_RULE	RW	0x150	Security access rules for AHB peripherals. Each peripheral can be assigned independent security level	0x0	47.4.40
SEC_CTRL_USB_HS_SLAVE_RULE	RW	0x160	This register has the security access rules for the slave port P11 on AHB multilayer. This slave port allows access to USB-HS RAM memories.	0x0	47.4.41

Table 888. Register overview: AHB_Secure_CTRL (base address = 0x500AC000) *...continued* [1] [2]

Name	Access	Offset	Description	Reset value	Section
SEC_CTRL_USB_HS_MEM_RULE	RW	0x170	Security access rules for USB-HS RAM sub region 0_0 to 0_3. Each USB-HS RAM sub region is 4 kbytes	0x0	47.4.42
SEC_VIO_ADDR0	R	0xE00	Most recent security violation address for AHB port ADDR0	0x0	47.4.43
SEC_VIO_ADDR1	R	0xE04	Most recent security violation address for AHB port ADDR1	0x0	47.4.44
SEC_VIO_ADDR2	R	0xE08	Most recent security violation address for AHB port ADDR2	0x0	47.4.45
SEC_VIO_ADDR3	R	0xE0C	Most recent security violation address for AHB port ADDR3	0x0	47.4.46
SEC_VIO_ADDR4	R	0xE10	Most recent security violation address for AHB port ADDR4	0x0	47.4.47
SEC_VIO_ADDR5	R	0xE14	Most recent security violation address for AHB port ADDR5	0x0	47.4.48
SEC_VIO_ADDR6	R	0xE18	Most recent security violation address for AHB port ADDR6	0x0	47.4.49
SEC_VIO_ADDR7	R	0xE1C	Most recent security violation address for AHB port ADDR7	0x0	47.4.50
SEC_VIO_ADDR8	R	0xE20	Most recent security violation address for AHB port ADDR8	0x0	47.4.51
SEC_VIO_ADDR9	R	0xE24	Most recent security violation address for AHB port ADDR9	0x0	47.4.52
SEC_VIO_ADDR10	R	0xE28	Most recent security violation address for AHB port ADDR10	0x0	47.4.53
SEC_VIO_ADDR11	R	0xE2C	Most recent security violation address for AHB port ADDR11	0x0	47.4.54
SEC_VIO_MISC_INFO0	R	0xE80	Most recent security violation miscellaneous information for AHB port INFO0	0x0	47.4.55
SEC_VIO_MISC_INFO1	R	0xE84	Most recent security violation miscellaneous information for AHB port INFO1	0x0	47.4.56
SEC_VIO_MISC_INFO2	R	0xE88	Most recent security violation miscellaneous information for AHB port INFO2	0x0	47.4.57
SEC_VIO_MISC_INFO3	R	0xE8C	Most recent security violation miscellaneous information for AHB port INFO3	0x0	47.4.58
SEC_VIO_MISC_INFO4	R	0xE90	Most recent security violation miscellaneous information for AHB port INFO4	0x0	47.4.59
SEC_VIO_MISC_INFO5	R	0xE94	Most recent security violation miscellaneous information for AHB port INFO5	0x0	47.4.60
SEC_VIO_MISC_INFO6	R	0xE98	Most recent security violation miscellaneous information for AHB port INFO6	0x0	47.4.61

Table 888. Register overview: AHB_Secure_CTRL (base address = 0x500AC000) *...continued* [1] [2]

Name	Access	Offset	Description	Reset value	Section
SEC_VIO_MISC_INFO7	R	0xE9C	Most recent security violation miscellaneous information for AHB port INFO7	0x0	47.4.62
SEC_VIO_MISC_INFO8	R	0xEA0	most recent security violation miscellaneous information for AHB port INFO8	0x0	47.4.63
SEC_VIO_MISC_INFO9	R	0xEA4	Most recent security violation miscellaneous information for AHB port INFO9	0x0	47.4.64
SEC_VIO_MISC_INFO10	R	0xEA8	Most recent security violation miscellaneous information for AHB port INFO10	0x0	47.4.65
SEC_VIO_MISC_INFO11	R	0xEAC	Most recent security violation miscellaneous information for AHB port INFO11	0x0	47.4.66
SEC_VIO_INFO_VALID	RW	0xF00	security violation address/information registers valid flags	0x0	47.4.67
SEC_GPIO_MASK0	RW	0xF80	Secure GPIO mask for port 0 pins. This register is used to block leakage of Secure interface (GPIOs, I2C, UART, and other peripherals configured as Secure peripherals) pin states to Non-secure world by reading pin states from normal GPIO port. If this port is not masked, its port pin states can be read using normal GPIO port even if these port pins are configured as other digital functions (UART, I2C) than GPIO. This mask does not apply during power-down mode so that a secure GPIO can be used as a wakeup source through GINT0.	0xFFFFFFFF	47.4.68
SEC_GPIO_MASK1	RW	0xF84	Secure GPIO mask for port 1 pins.	0xFFFFFFFF	47.4.69
SEC_CPU_INT_MASK0	RW	0xF90	Secure Interrupt mask for CPU1.	0xFFFFFFFF	47.4.70
SEC_CPU_INT_MASK1	RW	0xF94	Secure Interrupt mask for CPU1.	0xFFFFFFFF	47.4.71
SEC_MASK_LOCK	RW	0xFBC	Security General Purpose register access control.	0x0	47.4.72
MASTER_SEC_LEVEL	RW	0xFD0	Master Secure level register	0x0	47.4.73
MASTER_SEC_ANTI_POL_REG	RW	0xFD4	Master Secure level anti-pole register.	0x0	47.4.74
CPU0_LOCK_REG	RW	0xFEC	Miscellaneous control signals for in CPU0.	0x0	47.4.75

Table 888. Register overview: AHB_Secure_CTRL (base address = 0x500AC000) *...continued* [1] [2]

Name	Access	Offset	Description	Reset value	Section
CPU1_LOCK_REG	RW	0xFF0	Miscellaneous control signals for in CPU1.	0x0	47.4.76
MISC_CTRL_DP_REG	RW	0xFF8	Secure control duplicate register.	0x0	47.4.77
MISC_CTRL_REG	RW	0xFFC	Secure control register.	0x0	47.4.78

[1] Unlike other register tables, the base address noted for this function is the Secure address. This is because these registers are configured by Secure code and Non-secure accesses are denied.

[2] For all reserved registers and reserved bits within address range 0x500AC0F0 to 0x500AC174, value must be set to 1. See the SDK software platform for example code.

47.4.1 Security control Flash ROM slave rule

This register has the security access rules for the slave port P0 on AHB multilayer. This slave port allows access to Flash and ROM memories. This rule supersedes more granular security rules as in SEC_CTRL_FLASH_MEM_RULEx or SEC_CTRL_ROM_MEM_RULEx

Table 889. Security control Flash ROM slave rule (SEC_CTRL_FLASH_ROM_SLAVE_RULE, offset = 0x0)

Bit	Symbol	Access	Value	Description	Reset value
1:0	FLASH_RULE	RW		Security access rules for the whole FLASH: 0x0000_0000 - 0x0009_FFFF.	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved	0x0
5:4	ROM_RULE	RW		Security access rules for the whole ROM: 0x0300_0000 - 0x0301_FFFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:6		WO		Reserved.	0x0

Flash region is divided into 20 sub-regions, each rule corresponds to a following sub-region address range:

Flash rule 0_0 address space: 0x0000_0000 - 0x0000_7FFF

Flash rule 0_1 address space: 0x0000_8000 - 0x0000_FFFF

Flash rule 0_2 address space: 0x0001_0000 - 0x0001_7FFF

Flash rule 0_3 address space: 0x0001_8000 - 0x0001_FFFF

Flash rule 0_4 address space: 0x0002_0000 - 0x0002_7FFF

Flash rule 0_5 address space: 0x0002_8000 - 0x0002_FFFF

Flash rule 0_6 address space: 0x0003_0000 - 0x0003_7FFF

Flash rule 0_7 address space: 0x0003_8000 - 0x0003_FFFF

Flash rule 1_0 address space: 0x0004_0000 - 0x0004_7FFF

Flash rule 1_1 address space: 0x0004_8000 - 0x0004_FFFF

Flash rule 1_2 address space: 0x0005_0000 - 0x0005_7FFF

Flash rule 1_3 address space: 0x0005_8000 - 0x0005_FFFF

Flash rule 1_4 address space: 0x0006_0000 - 0x0006_7FFF

Flash rule 1_5 address space: 0x0006_8000 - 0x0006_FFFF

Flash rule 1_6 address space: 0x0007_0000 - 0x0007_7FFF

Flash rule 1_7 address space: 0x0007_8000 - 0x0007_FFFF

Flash rule 2_0 address space: 0x0008_0000 - 0x0008_7FFF

Flash rule 2_1 address space: 0x0008_8000 - 0x0008_FFFF

Flash rule 2_2 address space: 0x0009_0000 - 0x0009_7FFF

Flash rule 2_3 address space: 0x0009_8000 - 0x0009_FFFF

ROM region is divided into 32 sub-regions, each rule corresponds to a following sub-region address range:

Mem rule 0_0 address space: 0x0300_0000 - 0x0300_0FFF

Mem rule 0_1 address space: 0x0300_1000 - 0x0300_1FFF

Mem rule 0_2 address space: 0x0300_2000 - 0x0300_2FFF

Mem rule 0_3 address space: 0x0300_3000 - 0x0300_3FFF

Mem rule 0_4 address space: 0x0300_4000 - 0x0300_4FFF

Mem rule 0_5 address space: 0x0300_5000 - 0x0300_5FFF

Mem rule 0_6 address space: 0x0300_6000 - 0x0300_6FFF

Mem rule 0_7 address space: 0x0300_7000 - 0x0300_7FFF

Mem rule 1_0 address space: 0x0300_8000 - 0x0300_8FFF

Mem rule 1_1 address space: 0x0300_9000 - 0x0300_9FFF

Mem rule 1_2 address space: 0x0300_A000 - 0x0300_AFFF

Mem rule 1_3 address space: 0x0300_B000 - 0x0300_BFFF

Mem rule 1_4 address space: 0x0300_C000 - 0x0300_CFFF

Mem rule 1_5 address space: 0x0300_D000 - 0x0300_DFFF

Mem rule 1_6 address space: 0x0300_E000 - 0x0300_EFFF

Mem rule 1_7 address space: 0x0300_F000 - 0x0300_FFFF

Mem rule 2_0 address space: 0x0301_0000 - 0x0301_0FFF

Mem rule 2_1 address space: 0x0301_1000 - 0x0301_1FFF

Mem rule 2_2 address space: 0x0301_2000 - 0x0301_2FFF

Mem rule 2_3 address space: 0x0301_3000 - 0x0301_3FFF

Mem rule 2_4 address space: 0x0301_4000 - 0x0301_4FFF

Mem rule 2_5 address space: 0x0301_5000 - 0x0301_5FFF

Mem rule 2_6 address space: 0x0301_6000 - 0x0301_6FFF

Mem rule 2_7 address space: 0x0301_7000 - 0x0301_7FFF

Mem rule 3_0 address space: 0x0301_8000 - 0x0301_8FFF

Mem rule 3_1 address space: 0x0301_9000 - 0x0301_9FFF

Mem rule 3_2 address space: 0x0301_A000 - 0x0301_AFFF

Mem rule 3_3 address space: 0x0301_B000 - 0x0301_BFFF

Mem rule 3_4 address space: 0x0301_C000 - 0x0301_CFFF

Mem rule 3_5 address space: 0x0301_D000 - 0x0301_DFFF

Mem rule 3_6 address space: 0x0301_E000 - 0x0301_EFFF

Mem rule 3_7 address space: 0x0301_F000 - 0x0301_FFFF

47.4.2 Security control flash memory rule 0 register

This register has the security access rules for FLASH sector 0 to sector 20. Each flash sector is 32 kbytes. There are 20 FLASH sectors in total.

Table 890. Security control flash memory rule0 (SEC_CTRL_FLASH_MEM_RULE0, offset = 0x10)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0

Table 890. Security control flash memory rule0 (SEC_CTRL_FLASH_MEM_RULE0, offset = 0x10) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
5:4	RULE1	RW		Secure control rule1. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved	0x0
9:8	RULE2	RW		Secure control rule2. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved	0x0
13:12	RULE3	RW		Secure control rule3. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0
17:16	RULE4	RW		Secure control rule4. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved	0x0
21:20	RULE5	RW		Secure control rule5. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0
25:24	RULE6	RW		Secure control rule6. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved.	0x0

Table 890. Security control flash memory rule0 (SEC_CTRL_FLASH_MEM_RULE0, offset = 0x10) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
29:28	RULE7	RW		Secure control rule7. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.3 Security control flash memory rule 1 register

This registers has security access rules for FLASH sector 8 to sector 15. Each Flash sector is 32 kbytes.There are 20 FLASH sectors in total.

Table 891. Security control flash memory rule1 (SEC_CTRL_FLASH_MEM_RULE1, offset = 0x14)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved	0x0
5:4	RULE1	RW		Secure control rule1. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved	0x0
9:8	RULE2	RW		Secure control rule2. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved	0x0
13:12	RULE3	RW		Secure control rule3. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved	0x0
17:16	RULE4	RW		Secure control rule4. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	

Table 891. Security control flash memory rule1 (SEC_CTRL_FLASH_MEM_RULE1, offset = 0x14) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
19:18		WO		Reserved	0x0
21:20	RULE5	RW		Secure control rule5. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved	0x0
25:24	RULE6	RW		Secure control rule6. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved	0x0
29:28	RULE7	RW		Secure control rule7. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved	0x0

47.4.4 Security control flash memory rule 2 register

Security access rules for flash sector 6 to sector 19. Each Flash sector is 32 kbytes.There are 20 flash sectors in total.

Table 892. Security control flash memory rule2 register (SEC_CTRL_FLASH_MEM_RULE2, offset = 0x18)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved	0x0
5:4	RULE1	RW		Secure control rule1. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved	0x0

Table 892. Security control flash memory rule2 register (SEC_CTRL_FLASH_MEM_RULE2, offset = 0x18) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
9:8	RULE2	RW		Secure control rule2. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved	0x0
13:12	RULE3	RW		Secure control rule3. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved	0x0
17:16	RULE4	RW		Secure control rule4. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved	0x0
21:20	RULE5	RW		Secure control rule5. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved	0x0
25:24	RULE6	RW		Secure control rule6. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved	0x0
29:28	RULE7	RW		Secure control rule7. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved	0x0

47.4.5 Security control ROM memory rule 0 register

Security access rules for ROM sector 0 to sector 7. Each ROM sector is 4 kbytes.There are 32 ROM sectors in total.

Table 893. Security access rules for ROM (SEC_CTRL_ROM_MEM_RULE0, offset = 0x20)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	RULE1	RW		Secure control rule1. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	RULE2	RW		Secure control rule2. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	RULE3	RW		Secure control rule3. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0
17:16	RULE4	RW		Secure control rule4. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	RULE5	RW		Secure control rule5. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0
25:24	RULE6	RW		Secure control rule6. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	

Table 893. Security access rules for ROM (SEC_CTRL_ROM_MEM_RULE0, offset = 0x20) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
27:26		WO		Reserved.	0x0
29:28	RULE7	RW		Secure control rule7. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.6 Security control ROM memory rule 1 register

Security access rules for ROM sector 8 to sector 15. Each ROM sector is 4 kbytes.There are 32 ROM sectors in total.

Table 894. Security access rules for ROM sector 0 to sector 31(SEC_CTRL_ROM_MEM_RULE1, offset = 0x24)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	RULE1	RW		Secure control rule1. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	RULE2	RW		Secure control rule2. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	RULE3	RW		Secure control rule3. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0

Table 894. Security access rules for ROM sector 0 to sector 31(SEC_CTRL_ROM_MEM_RULE1, offset = 0x24)

Bit	Symbol	Access	Value	Description	Reset value
17:16	RULE4	RW		Secure control rule4. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	RULE5	RW		Secure control rule5. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0
25:24	RULE6	RW		Secure control rule6. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved.	0x0
29:28	RULE7	RW		Secure control rule7. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.7 Security control ROM memory rule 2 register

Security access rules for ROM sector 16 to sector 23. Each ROM sector is 4 kbytes.There are 32 ROM sectors in total

Table 895. Security access rules for ROM sector 0 to sector 31 (SEC_CTRL_ROM_MEM_RULE2, offset = 0x28)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	RULE1	RW		Secure control rule1. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	

Table 895. Security access rules for ROM sector 0 to sector 31 (SEC_CTRL_ROM_MEM_RULE2, offset = 0x28)

Bit	Symbol	Access	Value	Description	Reset value
7:6		WO		Reserved.	0x0
9:8	RULE2	RW		Secure control rule2. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	RULE3	RW		Secure control rule3. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0
17:16	RULE4	RW		Secure control rule4. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	RULE5	RW		Secure control rule5. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0
25:24	RULE6	RW		Secure control rule6. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved.	0x0
29:28	RULE7	RW		Secure control rule7. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.8 Security control ROM memory rule 3 register

Security access rules for ROM sector 24 to sector 31. Each ROM sector is 4 kbytes.There are 32 ROM sectors in total.

Table 896. Security access rules for ROM sector 0 to sector 31 (SEC_CTRL_ROM_MEM_RULE3, offset = 0x2C)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	RULE1	RW		Secure control rule1. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	RULE2	RW		Secure control rule2. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	RULE3	RW		Secure control rule3. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved	0x0
17:16	RULE4	RW		Secure control rule4. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	RULE5	RW		Secure control rule5. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0
25:24	RULE6	RW		Secure control rule6. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	

Table 896. Security access rules for ROM sector 0 to sector 31 (SEC_CTRL_ROM_MEM_RULE3, offset = 0x2C)

Bit	Symbol	Access	Value	Description	Reset value
27:26		WO		Reserved.	0x0
29:28	RULE7	RW		Secure control rule7. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.9 Security access rules for RAMX slaves

This register has the security access rules for the slave port P1 on AHB multilayer. This slave port allows access to RAM-X memories. This rule supersedes more granular security rules as in SEC_CTRL_RAMX_MEM_RULEx

Table 897. Security access rules for RAMX slaves (SEC_CTRL_RAMX_SLAVE_RULE, offset = 0x30)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RAMX_RULE	RW		Security access rules for the whole RAMX : 0x0400_0000 - 0x0400_7FFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:2		WO		Reserved.	0x0

RAMX region is divided into 8 sub-regions, each rule corresponds to a following sub-region address range:

Mem rule 0_0 Address space: 0x0400_0000 - 0x0400_0FFF

Mem rule 0_1 Address space: 0x0400_1000 - 0x0400_1FFF

Mem rule 0_2 Address space: 0x0400_2000 - 0x0400_2FFF

Mem rule 0_3 Address space: 0x0400_3000 - 0x0400_3FFF

Mem rule 0_4 Address space: 0x0400_4000 - 0x0400_4FFF

Mem rule 0_5 Address space: 0x0400_5000 - 0x0400_5FFF

Mem rule 0_6 Address space: 0x0400_6000 - 0x0400_6FFF

Mem rule 0_7 Address space: 0x0400_7000 - 0x0400_7FFF

47.4.10 Security access rules for RAMX slaves

Security access rules for RAMX sub region 0_0 to 0_7. Each RAMX sub region is 4 kbytes.

Set CASPER and SRAM-X_0 SRAM-X_1 at same security attribute even if CASPER is not used because CASPER has access to SRAMX_0 and SRAMX_1

Table 898. Security access rules for RAMX slaves (SEC_CTRL_RAMX_MEM_RULE0, offset = 0x40)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	RULE1	RW		Secure control rule1. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	RULE2	RW		Secure control rule2. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	RULE3	RW		Secure control rule3. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0
17:16	RULE4	RW		Secure control rule4. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	RULE5	RW		Secure control rule5. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0

Table 898. Security access rules for RAMX slaves (SEC_CTRL_RAMX_MEM_RULE0, offset = 0x40) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
25:24	RULE6	RW		Secure control rule6. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved.	0x0
29:28	RULE7	RW		Secure control rule7. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.11 Security access rules for RAM0 slaves

This register has the security access rules for the slave port P2 on AHB multilayer. This slave port allows access to RAM0 memories. This rule supersedes more granular security rules as in SEC_CTRL_RAM0_MEM_RULEx.

Table 899. Security access rules for RAM0 slaves (SEC_CTRL_RAM0_SLAVE_RULE, offset = 0x50)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RAM0_RULE	RW		Security access rules for the whole RAM0 : 0x2000_0000 - 0x2000_FFFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:2		WO		Reserved.	0x0

RAM0 region is divided into 16 sub-regions, each rule corresponds to a following sub-region address range:

Mem rule 0_0 Address space: 0x2000_0000 - 0x2000_0FFF

Mem rule 0_1 Address space: 0x2000_1000 - 0x2000_1FFF

Mem rule 0_2 Address space: 0x2000_2000 - 0x2000_2FFF

Mem rule 0_3 Address space: 0x2000_3000 - 0x2000_3FFF

Mem rule 0_4 Address space: 0x2000_4000 - 0x2000_4FFF

Mem rule 0_5 Address space: 0x2000_5000 - 0x2000_5FFF

Mem rule 0_6 Address space: 0x2000_6000 - 0x2000_6FFF

Mem rule 0_7 Address space: 0x2000_7000 - 0x2000_7FFF

Mem rule 1_0 Address space: 0x2000_8000 - 0x2000_8FFF

Mem rule 1_1 Address space: 0x2000_9000 - 0x2000_9FFF

Mem rule 1_2 Address space: 0x2000_A000 - 0x2000_AFFF

Mem rule 1_3 Address space: 0x2000_B000 - 0x2000_BFFF

Mem rule 1_4 Address space: 0x2000_C000 - 0x2000_CFFF

Mem rule 1_5 Address space: 0x2000_D000 - 0x2000_DFFF

Mem rule 1_6 Address space: 0x2000_E000 - 0x2000_EFFF

Mem rule 1_7 Address space: 0x2000_F000 - 0x2000_FFFF

47.4.12 Security access rules for RAM0 slaves

Security access rules for RAM0 sub region 0_0 to 0_7. Each RAM0 sub region is 4 kbytes.

Table 900. Security access rules for RAM0 slaves (SEC_CTRL_RAM0_MEM_RULE0, offset = 0x60)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	RULE1	RW		Secure control rule1. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	RULE2	RW		Secure control rule2. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	RULE3	RW		Secure control rule3. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0

Table 900. Security access rules for RAM0 slaves (SEC_CTRL_RAM0_MEM_RULE0, offset = 0x60) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
17:16	RULE4	RW		Secure control rule4. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	RULE5	RW		Secure control rule5. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0
25:24	RULE6	RW		Secure control rule6. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved.	0x0
29:28	RULE7	RW		Secure control rule7. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.13 Security access rules for RAM0 slaves

Security access rules for RAM0 sub region 1_0 to 1_7. Each RAM0 sub region is 4 kbytes.

Table 901. Security access rules for RAM0 slaves (SEC_CTRL_RAM0_MEM_RULE1, offset = 0x64)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	RULE1	RW		Secure control rule1. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	

Table 901. Security access rules for RAM0 slaves (SEC_CTRL_RAM0_MEM_RULE1, offset = 0x64) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
7:6		WO		Reserved.	0x0
9:8	RULE2	RW		Secure control rule2. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	RULE3	RW		Secure control rule3. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0
17:16	RULE4	RW		Secure control rule4. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	RULE5	RW		Secure control rule5. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0
25:24	RULE6	RW		Secure control rule6. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved.	0x0
29:28	RULE7	RW		Secure control rule7. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.14 Security access rules for RAM1 slaves

This register has the security access rules for the slave port P3 on AHB multilayer. This slave port allows access to RAM1 memories. This rule supersedes more granular security rules as in SEC_CTRL_RAM1_MEM_RULEx

Table 902. Security access rules for RAM1 slaves (SEC_CTRL_RAM1_SLAVE_RULE, offset = 0x70)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RAM1_RULE	RW		Security access rules for the whole RAM1 : 0x2001_0000 - 0x2001_FFFF name=0	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:2		WO		Reserved.	0x0

RAM1 region is divided into 16 sub-regions, each rule corresponds to a following sub-region address range:

Mem rule 0_0 Address space: 0x2001_0000 - 0x2001_0FFF

Mem rule 0_1 Address space: 0x2001_1000 - 0x2001_1FFF

Mem rule 0_2 Address space: 0x2001_2000 - 0x2001_2FFF

Mem rule 0_3 Address space: 0x2001_3000 - 0x2001_3FFF

Mem rule 0_4 Address space: 0x2001_4000 - 0x2001_4FFF

Mem rule 0_5 Address space: 0x2001_5000 - 0x2001_5FFF

Mem rule 0_6 Address space: 0x2001_6000 - 0x2001_6FFF

Mem rule 0_7 Address space: 0x2001_7000 - 0x2001_7FFF

Mem rule 1_0 Address space: 0x2001_8000 - 0x2001_8FFF

Mem rule 1_1 Address space: 0x2001_9000 - 0x2001_9FFF

Mem rule 1_2 Address space: 0x2001_A000 - 0x2001_AFFF

Mem rule 1_3 Address space: 0x2001_B000 - 0x2001_BFFF

Mem rule 1_4 Address space: 0x2001_C000 - 0x2001_CFFF

Mem rule 1_5 Address space: 0x2001_D000 - 0x2001_DFFF

Mem rule 1_6 Address space: 0x2001_E000 - 0x2001_EFFF

Mem rule 1_7 Address space: 0x2001_F000 - 0x2001_FFFF

47.4.15 Security access rules for RAM1 slaves

Security access rules for RAM1 sub region 0_0 to 0_7. Each RAM1 sub region is 4 kbytes.

Table 903. Security access rules for RAM1 slaves (SEC_CTRL_RAM1_MEM_RULE0, offset = 0x80)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	RULE1	RW		Secure control rule1. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	RULE2	RW		Secure control rule2. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	RULE3	RW		Secure control rule3. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0
17:16	RULE4	RW		Secure control rule4. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	RULE5	RW		Secure control rule5. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0

Table 903. Security access rules for RAM1 slaves (SEC_CTRL_RAM1_MEM_RULE0, offset = 0x80) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
25:24	RULE6	RW		Secure control rule6. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved.	0x0
29:28	RULE7	RW		Secure control rule7. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.16 Security access rules for RAM1 slaves

Security access rules for RAM1 sub region 1_0 to 1_7. Each RAM1 sub region is 4 kbytes.

Table 904. Security access rules for RAM1 slaves (SEC_CTRL_RAM1_MEM_RULE1, offset = 0x84)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	RULE1	RW		Secure control rule1. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	RULE2	RW		Secure control rule2. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	RULE3	RW		Secure control rule3. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	

Table 904. Security access rules for RAM1 slaves (SEC_CTRL_RAM1_MEM_RULE1, offset = 0x84) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
15:14		WO		Reserved.	0x0
17:16	RULE4	RW		Secure control rule4. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	RULE5	RW		Secure control rule5. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0
25:24	RULE6	RW		Secure control rule6. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved.	0x0
29:28	RULE7	RW		Secure control rule7. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.17 Security access rules for RAM2 slaves

This register has the security access rules for the slave port P4 on AHB multilayer. This slave port allows access to RAM2 memories. This rule supersedes more granular security rules as in SEC_CTRL_RAM2_MEM_RULEx.

Table 905. Security access rules for RAM2 slaves (SEC_CTRL_RAM2_SLAVE_RULE, offset = 0x90)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RAM2_RULE	RW		Security access rules for the whole RAM2 : 0x2002_0000 - 0x2002_FFFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:2		WO		Reserved.	0x0

RAM2 region is divided into 16 sub-regions, each rule corresponds to a following sub-region address range:

Mem rule 0_0 Address space: 0x2002_0000 - 0x2002_0FFF

Mem rule 0_1 Address space: 0x2002_1000 - 0x2002_1FFF

Mem rule 0_2 Address space: 0x2002_2000 - 0x2002_2FFF

Mem rule 0_3 Address space: 0x2002_3000 - 0x2002_3FFF

Mem rule 0_4 Address space: 0x2002_4000 - 0x2002_4FFF

Mem rule 0_5 Address space: 0x2002_5000 - 0x2002_5FFF

Mem rule 0_6 Address space: 0x2002_6000 - 0x2002_6FFF

Mem rule 0_7 Address space: 0x2002_7000 - 0x2002_7FFF

Mem rule 1_0 Address space: 0x2002_8000 - 0x2002_8FFF

Mem rule 1_1 Address space: 0x2002_9000 - 0x2002_9FFF

Mem rule 1_2 Address space: 0x2002_A000 - 0x2002_AFFF

Mem rule 1_3 Address space: 0x2002_B000 - 0x2002_BFFF

Mem rule 1_4 Address space: 0x2002_C000 - 0x2002_CFFF

Mem rule 1_5 Address space: 0x2002_D000 - 0x2002_DFFF

Mem rule 1_6 Address space: 0x2002_E000 - 0x2002_EFFF

Mem rule 1_7 Address space: 0x2002_F000 - 0x2002_FFFF

47.4.18 Security access rules for RAM2 slaves

Security access rules for RAM2 sub region 0_0 to 0_7. Each RAM2 sub region is 4 kbytes.

Table 906. Security access rules for RAM2 slaves (SEC_CTRL_RAM2_MEM_RULE0, offset = 0xA0)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	RULE1	RW		Secure control rule1. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	RULE2	RW		Secure control rule2. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	RULE3	RW		Secure control rule3. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0
17:16	RULE4	RW		Secure control rule4. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	RULE5	RW		Secure control rule5. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0

Table 906. Security access rules for RAM2 slaves (SEC_CTRL_RAM2_MEM_RULE0, offset = 0xA0) *...continued*

Bit	Symbol	Access	Value	Description	Reset value
25:24	RULE6	RW		Secure control rule6. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved.	0x0
29:28	RULE7	RW		Secure control rule7. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.19 Security access rules for RAM2 slaves

Security access rules for RAM2 sub region 1_0 to 1_7. Each RAM2 sub region is 4 kbytes

Table 907. Security access rules for RAM2 slaves (SEC_CTRL_RAM2_MEM_RULE1, offset = 0xA4)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	RULE1	RW		Secure control rule1. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	RULE2	RW		Secure control rule2. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	RULE3	RW		Secure control rule3. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0
17:16	RULE4	RW		Secure control rule4. it can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	RULE5	RW		Secure control rule5. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0

Table 907. Security access rules for RAM2 slaves (SEC_CTRL_RAM2_MEM_RULE1, offset = 0xA4) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
25:24	RULE6	RW		Secure control rule6. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved.	0x0
29:28	RULE7	RW		Secure control rule7. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.20 Security access rules for RAM3 slaves

This register has the security access rules for the slave port P5 on AHB multilayer. This slave port allows access to RAM3 memories. This rule supersedes more granular security rules as in SEC_CTRL_RAM3_MEM_RULEx

Table 908. Security access rules for RAM3 slaves. (SEC_CTRL_RAM3_SLAVE_RULE, offset = 0xB0)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RAM3_RULE	RW		Security access rules for the whole RAM3: 0x2003_0000 - 0x2003_FFFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:2		WO		Reserved.	0x0

47.4.21 Security access rules for RAM3 slaves

Security access rules for RAM3 sub region 0_0 to 0_7. Each RAM3 sub region is 4 kbytes.

Table 909. Security access rules for RAM3 slaves(SEC_CTRL_RAM3_MEM_RULE0, offset = 0xC0)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0

Table 909. Security access rules for RAM3 slaves(SEC_CTRL_RAM3_MEM_RULE0, offset = 0xC0) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
5:4	RULE1	RW		Secure control rule1. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	RULE2	RW		Secure control rule2. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	RULE3	RW		Secure control rule3. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0
17:16	RULE4	RW		Secure control rule4. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	RULE5	RW		Secure control rule5. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0
25:24	RULE6	RW		Secure control rule6. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved.	0x0

Table 909. Security access rules for RAM3 slaves(SEC_CTRL_RAM3_MEM_RULE0, offset = 0xC0) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
29:28	RULE7	RW		Secure control rule7. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

RAM3 region is divided into 16 sub-regions, each rule corresponds to a following sub-region address range:

Mem rule 0_0 Address space: 0x2003_0000 - 0x2003_0FFF

Mem rule 0_1 Address space: 0x2003_1000 - 0x2003_1FFF

Mem rule 0_2 Address space: 0x2003_2000 - 0x2003_2FFF

Mem rule 0_3 Address space: 0x2003_3000 - 0x2003_3FFF

Mem rule 0_4 Address space: 0x2003_4000 - 0x2003_4FFF

Mem rule 0_5 Address space: 0x2003_5000 - 0x2003_5FFF

Mem rule 0_6 Address space: 0x2003_6000 - 0x2003_6FFF

Mem rule 0_7 Address space: 0x2003_7000 - 0x2003_7FFF

Mem rule 1_0 Address space: 0x2003_8000 - 0x2003_8FFF

Mem rule 1_1 Address space: 0x2003_9000 - 0x2003_9FFF

Mem rule 1_2 Address space: 0x2003_A000 - 0x2003_AFFF

Mem rule 1_3 Address space: 0x2003_B000 - 0x2003_BFFF

Mem rule 1_4 Address space: 0x2003_C000 - 0x2003_CFFF

Mem rule 1_5 Address space: 0x2003_D000 - 0x2003_DFFF

Mem rule 1_6 Address space: 0x2003_E000 - 0x2003_EFFF

Mem rule 1_7 Address space: 0x2003_F000 - 0x2003_FFFF

47.4.22 Security access rules for RAM3 slaves

Security access rules for RAM3 sub region 1_0 to 1_7. Each RAM3 sub region is 4 kbytes

Table 910. Security access rules for RAM3 slaves (SEC_CTRL_RAM3_MEM_RULE1, offset = 0xC4)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	RULE1	RW		Secure control rule1. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	RULE2	RW		Secure control rule2. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	RULE3	RW		Secure control rule3. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0
17:16	RULE4	RW		Secure control rule4. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	RULE5	RW		Secure control rule5. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0
25:24	RULE6	RW		Secure control rule6. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	

Table 910. Security access rules for RAM3 slaves (SEC_CTRL_RAM3_MEM_RULE1, offset = 0xC4) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
27:26		WO		Reserved.	0x0
29:28	RULE7	RW		Secure control rule7. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.23 Security access rules for RAM4 slaves

This register has the security access rules for the slave port P6 on AHB multilayer. This slave port allows access to RAM3 memories. This rule supersedes more granular security rules as in SEC_CTRL_RAM4_MEM_RULEx

Table 911. Security access rules for RAM4 slaves. (SEC_CTRL_RAM4_SLAVE_RULE, offset = 0xD0)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RAM4_RULE	RW		Security access rules for the whole RAM4 : 0x2004_0000 - 0x2004_3FFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:2		WO		Reserved.	0x0

RAM4 region is divided into 4 sub-regions, each rule corresponds to a following sub-region address range:

Mem rule 0_0 Address space: 0x2004_0000 - 0x2004_0FFF

Mem rule 0_1 Address space: 0x2004_1000 - 0x2004_1FFF

Mem rule 0_2 Address space: 0x2004_2000 - 0x2004_2FFF

Mem rule 0_3 Address space: 0x2004_3000 - 0x2004_3FFF

47.4.24 Security access rules for RAM4 slaves

Set CASPER and SRAM4 at security attribute even if POWERQUAD is not used because POWERQUAD has access to SRAM4_0/1/2/3.

Table 912. Security access rules for RAM4 slaves (SEC_CTRL_RAM4_MEM_RULE0, offset = 0xE0)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RULE0	RW		Secure control rule0. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0

Table 912. Security access rules for RAM4 slaves (SEC_CTRL_RAM4_MEM_RULE0, offset = 0xE0) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
5:4	RULE1	RW		Secure control rule1. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	RULE2	RW		Secure control rule2. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	RULE3	RW		Secure control rule3. It can be set when check_reg's write_lock is '0'	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:14		WO		Reserved.	0x0

47.4.25 Security control APB bridge slave rule

This register has the security access rules for the slave port P7 on AHB multilayer. This slave port allows access to peripherals on two APB Bridges. This rule supersedes more granular security rules as in SEC_CTRL_APB_BRIDGE0_MEM_CTRLx.

Table 913. Security control APB bridge slave rule (SEC_CTRL_APB_BRIDGE_SLAVE_RULE, offset = 0xF0)

Bit	Symbol	Access	Value	Description	Reset value
1:0	APBBRIDGE0_RULE	RW		Security access rules for the whole APB Bridge 0	0x0
			0	Non-secure and Non-privilege user access allowed.	
			1	Non-secure and Privilege access allowed.	
			2	Secure and Non-privilege user access allowed.	
			3	Secure and privilege user access allowed.	
3:2		WO		Reserved.	0x0
5:4	APBBRIDGE1_RULE	RW		Security access rules for the whole APB Bridge 1	0x0
			0	Non-secure and Non-privilege user access allowed.	
			1	Non-secure and Privilege access allowed.	
			2	Secure and Non-privilege user access allowed.	
			3	Secure and privilege user access allowed.	
31:6		WO		Reserved.	0x0

47.4.26 Secure control APB bridge0 memory control0

Security access rules for APB Bridge 0 peripherals. Each peripheral can have independent security attribute

Table 914. Secure control APB Bridge0 memory control0 (SEC_CTRL_APB_BRIDGE0_MEM_CTRL0, offset = 0x100)

Bit	Symbol	Access	Value	Description	Reset value
1:0	SYSCON_RULE	RW		System configuration	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	IOCON_RULE	RW		I/O configuration	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	GINT0_RULE	RW		GPIO input Interrupt 0	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	GINT1_RULE	RW		GPIO input Interrupt 1	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0
17:16	PINT_RULE	RW		Pin Interrupt and Pattern match	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	SEC_PINT_RULE	RW		Secure Pin Interrupt and Pattern match	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0

Table 914. Secure control APB Bridge0 memory control0 (SEC_CTRL_APB_BRIDGE0_MEM_CTRL0, offset = 0x100)

Bit	Symbol	Access	Value	Description	Reset value
25:24	INPUTMUX_RULE	RW		Peripheral Input Multiplexing.	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:26		WO		Reserved.	0x0

47.4.27 Secure control APB bridge0 memory control1

Security access rules for APB Bridge 0 peripherals. Each peripheral can have independent security attribute.

Table 915. Secure control APB Bridge0 memory control1 (SEC_CTRL_APB_BRIDGE0_MEM_CTRL1, offset = 0x104)

Bit	Symbol	Access	Value	Description	Reset value
1:0	CTIMER0_RULE	RW		Standard counter/Timer 0	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	CTIMER1_RULE	RW		Standard counter/Timer 1	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:6		WO		Reserved.	0x0
17:16	WWDT_RULE	RW		Windowed watchdog Timer	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	MRT_RULE	RW		Multi-rate Timer	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0

Table 915. Secure control APB Bridge0 memory control1 (SEC_CTRL_APB_BRIDGE0_MEM_CTRL1, offset = 0x104)

…continued

Bit	Symbol	Access	Value	Description	Reset value
25:24	UTICK_RULE	RW		Micro-Timer	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:26		WO		Reserved.	0x0

47.4.28 Secure control APB bridge0 memory control2

Security access rules for APB Bridge 0 peripherals. Each peripheral can have independent security attribute.

Table 916. Secure control APB Bridge0 memory control2 (SEC_CTRL_APB_BRIDGE0_MEM_CTRL2, offset = 0x108)

Bit	Symbol	Access	Value	Description	Reset value
11:0		WO		Reserved.	0x0
13:12	ANACTRL_RULE	RW		Analog modules controller	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:14		WO		Reserved.	0x0

47.4.29 Secure control APB bridge0 memory control 3

Security access rules for APB Bridge 0 peripherals. Each peripheral can have independent security attribute

47.4.30 Secure control APB bridge1 memory control 0

Security access rules for APB Bridge 1 peripherals. Each peripheral can have independent security attribute.

Table 917. Secure control APB Bridge1 memory control 0(SEC_CTRL_APB_BRIDGE1_MEM_CTRL0, offset = 0x110)

Bit	Symbol	Access	Value	Description	Reset value
1:0	PMC_RULE	RW		Power Management Controller	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:2		WO		Reserved.	0x0

Table 917. Secure control APB Bridge1 memory control 0(SEC_CTRL_APB_BRIDGE1_MEM_CTRL0, offset = 0x110)

Bit	Symbol	Access	Value	Description	Reset value
13:12	SYSCTRL_RULE	RW		System Controller	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:14		WO		Reserved.	0x0

47.4.31 Secure control APB bridge1 memory control1

Security access rules for APB Bridge 1 peripherals. Each peripheral can have independent security attribute.

Table 918. Secure Control APB Bridge1 Memory Control1 (SEC_CTRL_APB_BRIDGE1_MEM_CTRL1, offset = 0x114)

Bit	Symbol	Access	Value	Description	Reset value
1:0	CTIMER2_RULE	RW		Standard counter/Timer 2	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	CTIMER3_RULE	RW		Standard counter/Timer 3	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	CTIMER4_RULE	RW		Standard counter/Timer 4	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:10		WO		Reserved.	0x0
17:16	RTC_RULE	RW		Real Time Counter	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0

Table 918. Secure Control APB Bridge1 Memory Control1 (SEC_CTRL_APB_BRIDGE1_MEM_CTRL1, offset = 0x114) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
21:20	OSEVENT_RULE	RW		OS Event Timer	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:22		WO		Reserved.	0x0

47.4.32 Secure control APB bridge1 memory control2 register

Security access rules for APB Bridge 1 peripherals. Each peripheral can have independent security attribute.

Table 919. Secure control APB bridge1 memory control2 (SEC_CTRL_APB_BRIDGE1_MEM_CTRL2, offset = 0x118)

Bit	Symbol	Access	Value	Description	Reset value
15:0		WO		Reserved.	0x0
17:16	FLASH_CTRL_RULE	RW		Flash controller	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	PRINCE_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:22		WO		Reserved.	0x0

47.4.33 Security access rules for APB Bridge 1 peripherals.

Security access rules for APB Bridge 1 peripherals. Each peripheral can have independent security attribute.

Table 920. Secure control APB bridge1 memory control3 register (SEC_CTRL_APB_BRIDGE1_MEM_CTRL3, offset = 0x11C)

Bit	Symbol	Access	Value	Description	Reset value
1:0	USBHPHY_RULE	RW		USB High Speed Phy controller	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:2		WO		Reserved.	0x0

Table 920. Secure control APB bridge1 memory control3 register (SEC_CTRL_APB_BRIDGE1_MEM_CTRL3, offset = 0x11C) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
9:8	RNG_RULE	RW		True Random Number Generator	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	PUF_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:14		WO		Reserved.	0x0
21:20	PLU_RULE	RW		Programmable Look-Up logic	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:22		WO		Reserved.	0x0

47.4.34 Security access rules for AHB peripherals

Security access rules for AHB peripherals on AHB Slave Port P8. Each peripheral can have independent security attribute.

Table 921. Security control AHB0 slave rule (SEC_CTRL_AHB_PORT8_SLAVE0_RULE, offset = 0x120)

Bit	Symbol	Access	Value	Description	Reset value
7:0		WO		Reserved.	0x0
9:8	DMA0_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:10		WO		Reserved.	0x0
17:16	FS_USB_DEV_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	SCT_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0
25:24	FLEXCOMM0_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved.	0x0
29:28	FLEXCOMM1_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.35 Security access rules for AHB peripherals

Security access rules for AHB peripherals on AHB Slave Port P8. Each peripheral can have independent security attribute.

Table 922. Security access rules for AHB peripherals. (SEC_CTRL_AHB_PORT8_SLAVE1_RULE, offset = 0x124)

Bit	Symbol	Access	Value	Description	Reset value
1:0	FLEXCOMM2_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	FLEXCOMM3_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	FLEXCOMM4_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	MAILBOX_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0
17:16	GPIO0_RULE	RW		High Speed GPIO	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:18		WO		Reserved.	0x0

47.4.36 Security access rules for AHB peripherals

Security access rules for AHB peripherals on AHB Slave Port P9. Each peripheral can have independent security attribute.

Table 923. Security access rules for AHB peripherals. (SEC_CTRL_AHB_PORT9_SLAVE0_RULE, offset =0x130)

Bit	Symbol	Access	Value	Description	Reset value
15:0		WO		Reserved.	0x0

Table 923. Security access rules for AHB peripherals. (SEC_CTRL_AHB_PORT9_SLAVE0_RULE, offset =0x130)

Bit	Symbol	Access	Value	Description	Reset value
17:16	USB_HS_DEV_RULE	RW		USB high Speed device registers	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	CRC_RULE	RW		CRC engine	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0
25:24	FLEXCOMM5_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved.	0x0
29:28	FLEXCOMM6_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.37 Security access rules for AHB peripherals

Security access rules for AHB peripherals on AHB Slave Port P9. Each peripheral can have independent security attribute.

Table 924. Security access rules for AHB peripherals. (SEC_CTRL_AHB_PORT9_SLAVE1_RULE, offset = 0x134)

Bit	Symbol	Access	Value	Description	Reset value
1:0	FLEXCOMM7_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:2		WO		Reserved.	0x0
13:12	SDIO_RULE	RW			0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	

Table 924. Security access rules for AHB peripherals. (SEC_CTRL_AHB_PORT9_SLAVE1_RULE, offset = 0x134)

Bit	Symbol	Access	Value	Description	Reset value
15:14		WO		Reserved.	0x0
17:16	DBG_MAILBOX_RULE	RW		Debug mailbox (aka ISP-AP)	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:18		WO		Reserved.	0x0
29:28	HS_LSPI_RULE	RW		High Speed SPI	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.38 Security access rules for AHB peripherals

Security access rules for AHB peripherals on AHB Slave Port P10. Each peripheral can have independent security attribute.

Table 925. Security access rules for AHB peripherals (SEC_CTRL_AHB_PORT10_SLAVE0_RULE, offset = 0x140)

Bit	Symbol	Access	Value	Description	Reset value
1:0	ADC_RULE	RW		ADC	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:2		WO		Reserved.	0x0
9:8	USB_FS_HOST_RULE	RW		USB Full Speed Host registers.	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	USB_HS_HOST_RULE	RW		USB High speed host registers	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
15:14		WO		Reserved.	0x0

Table 925. Security access rules for AHB peripherals (SEC_CTRL_AHB_PORT10_SLAVE0_RULE, offset = 0x140)

Bit	Symbol	Access	Value	Description	Reset value
17:16	HASH_RULE	RW		SHA-2 crypto registers	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
19:18		WO		Reserved.	0x0
21:20	CASPER_RULE	RW		RSA/ECC crypto accelerator	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
23:22		WO		Reserved.	0x0
25:24	PQ_RULE	RW		Power Quad (CM33 processor hardware accelerator)	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
27:26		WO		Reserved.	0x0
29:28	DMA1_RULE	RW		DMA Controller (Secure)	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:30		WO		Reserved.	0x0

47.4.39 Security access rules for AHB peripherals

Security access rules for AHB peripherals on AHB Slave Port P10. Each peripheral can have independent security attribute.

Table 926. Security access rules for AHB peripherals (SEC_CTRL_AHB_PORT10_SLAVE1_RULE, offset = 0x144)

Bit	Symbol	Access	Value	Description	Reset value
1:0	GPIO1_RULE	RW		Secure High Speed GPIO	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0

Table 926. Security access rules for AHB peripherals (SEC_CTRL_AHB_PORT10_SLAVE1_RULE, offset = 0x144)

Bit	Symbol	Access	Value	Description	Reset value
5:4	AHB_SEC_CTRL_RULE	RW		AHB Secure Controller	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:6		WO		Reserved.	0x0

47.4.40 Security control AHB2 memory rule

Security access rules for AHB secure control.

Write access attributes for this module are tier-4 (secure privileged). Fields below allow setting the read attributes with different tiers so that Masters with attributes other than tier-4 can read secure AHB Controller registers, if application wants to allow that. Base 4k region ie 0x4000_C000-0x4000_CFFF is mirrored at 0x4000_D000-0x4000_DFFF, 0x4000_E000-0x4000_EFFF, 0x4000_F000-0x4000_FFFF. It is applicable for read only, programing this register doesn't change write attribute to this module

Table 927. Security control AHB2 (SEC_CTRL_AHB2_0_MEM_RULE, offset = 0x150)

Bit	Symbol	Access	Value	Description	Reset value
1:0	AHB_SEC_CTRL_SECT_0_RULE	RW		Address space: 0x400A_0000 - 0x400A_CFFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0
5:4	AHB_SEC_CTRL_SECT_1_RULE	RW		Address space: 0x400A_D000 - 0x400A_DFFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	AHB_SEC_CTRL_SECT_2_RULE	RW		Address space: 0x400A_E000 - 0x400A_EFFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0

Table 927. Security control AHB2 (SEC_CTRL_AHB2_0_MEM_RULE, offset = 0x150) *...continued*

Bit	Symbol	Access	Value	Description	Reset value
13:12	AHB_SEC_CTRL_SECT_3_RULE	RW		Address space: 0x400A_F000 - 0x400A_FFFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:14		WO		Reserved.	0x0

47.4.41 Security control USB HS slave rule

This register has the security access rules for the slave port P11 on AHB multilayer. This slave port allows access to USB-HS RAM memories. This rule supersedes more granular security rules as in SEC_CTRL_USB_HS_SLAVE_RULE

Table 928. Security control USB slave rule (SEC_CTRL_USB_HS_SLAVE_RULE, offset = 0x160)

Bit	Symbol	Access	Value	Description	Reset value
1:0	RAM_USB_HS_RULE	RW		Security access rules for the entire USB high-speed RAM: 0x4010_0000 - 0x4010_3FFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:2		WO		Reserved.	0x0

USB-HS RAM region is divided into 4 sub-regions, each rule corresponds to a following sub-region address range:

Mem rule 0_0 Address space: 0x4010_0000 - 0x4010_0FFF

Mem rule 0_1 Address space: 0x4010_1000 - 0x4010_1FFF

Mem rule 0_2 Address space: 0x4010_2000 - 0x4010_2FFF

Mem rule 0_3 Address space: 0x4010_3000 - 0x4010_3FFF

47.4.42 Security control USB HS memory slave rule

Security access rules for USB-HS RAM sub region 0_0 to 0_3. Each USB-HS RAM sub region is 4 kbytes

Table 929. Security control USB HS slave rule (SEC_CTRL_USB_HS_MEM_RULE, offset = 0x170)

Bit	Symbol	Access	Value	Description	Reset value
1:0	SRAM_SECT_0_RULE	RW		Address space: 0x4010_0000 - 0x4010_0FFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
3:2		WO		Reserved.	0x0

Table 929. Security control USB HS slave rule (SEC_CTRL_USB_HS_MEM_RULE, offset = 0x170) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
5:4	SRAM_SECT_1_RULE	RW		Address space: 0x4010_1000 - 0x4010_1FFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
7:6		WO		Reserved.	0x0
9:8	SRAM_SECT_2_RULE	RW		Address space: 0x4010_2000 - 0x4010_2FFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
11:10		WO		Reserved.	0x0
13:12	SRAM_SECT_3_RULE	RW		Address space: 0x4010_3000 - 0x4010_3FFF	0x0
			0	Non-secure and non-privileged user access allowed.	
			1	Non-secure and privileged access allowed.	
			2	Secure and non-privileged user access allowed.	
			3	Secure and privileged user access allowed.	
31:14		WO		Reserved.	0x0

47.4.43 Security violation address for AHB port 0

This is the most recent security violation address for AHB port 0. However, please refer to SEC_VIO_INFO_VALID register to verify if that slave has valid violation.

Table 930. Security violation address for AHB port 0 (sec_vio_addr0, offset = 0xE00)

Bit	Symbol	Access	Description	Reset value
31:0	SEC_VIO_ADDR	RO	Security violation address for AHB port.	0x0

47.4.44 Security violation address for AHB port 1

This is the most recent security violation address for AHB port 1. However, please refer to SEC_VIO_INFO_VALID register to verify if that slave has valid violation.

Table 931. Security violation address for AHB port 1 (sec_vio_addr1, offset = 0xE04)

Bit	Symbol	Access	Description	Reset value
31:0	SEC_VIO_ADDR	RO	Security violation address for AHB port.	0x0

47.4.45 Security violation address for AHB port 2

This is the most recent security violation address for AHB port 2. However, please refer to SEC_VIO_INFO_VALID register to verify if that slave has valid violation.

Table 932. Security violation address for AHB port 2 (sec_vio_addr2, offset = 0xE08)

Bit	Symbol	Access	Description	Reset value
31:0	SEC_VIO_ADDR	RO	Security violation address for AHB port.	0x0

47.4.46 Security violation address for AHB port 3

This is the most recent security violation address for AHB port 3. However, please refer to SEC_VIO_INFO_VALID register to verify if that slave has valid violation.

Table 933. Security violation address for AHB port 3 (sec_vio_addr3, offset = 0xE0C)

Bit	Symbol	Access	Description	Reset value
31:0	SEC_VIO_ADDR	RO	Security violation address for AHB port.	0x0

47.4.47 Security violation address for AHB port 4

This is the most recent security violation address for AHB port 4. However, please refer to SEC_VIO_INFO_VALID register to verify if that slave has valid violation.

Table 934. Security violation address for AHB port 4 (sec_vio_addr4, offset = 0xE10)

Bit	Symbol	Access	Description	Reset value
31:0	SEC_VIO_ADDR	RO	Security violation address for AHB port.	0x0

47.4.48 Security violation address for AHB port 5

This is the most recent security violation address for AHB port 5. However, please refer to SEC_VIO_INFO_VALID register to verify if that slave has valid violation

Table 935. Security violation address for AHB port 5 (sec_vio_addr5, offset = 0xE14)

Bit	Symbol	Access	Description	Reset value
31:0	SEC_VIO_ADDR	RO	Security violation address for AHB port.	0x0

47.4.49 Security violation address for AHB port 6

This is the most recent security violation address for AHB port 6. However, please refer to SEC_VIO_INFO_VALID register to verify if that slave has valid violation.

Table 936. Security violation address for AHB port 6 (sec_vio_addr6, offset = 0xE18)

Bit	Symbol	Access	Description	Reset value
31:0	SEC_VIO_ADDR	RO	Security violation address for AHB port.	0x0

47.4.50 Security violation address for AHB port 7

This is the most recent security violation address for AHB port 7. However, please refer to SEC_VIO_INFO_VALID register to verify if that slave has valid violation.

Table 937. Security violation address for AHB port 7 (sec_vio_addr7, offset = 0xE1C)

Bit	Symbol	Access	Description	Reset value
31:0	SEC_VIO_ADDR	RO	Security violation address for AHB port.	0x0

47.4.51 Security violation address for AHB port 8

This is the most recent security violation address for AHB port 8. However, please refer to SEC_VIO_INFO_VALID register to verify if that slave has valid violation.

Table 938. Security violation address for AHB port 8 (sec_vio_addr8, offset = 0xE20)

Bit	Symbol	Access	Description	Reset value
31:0	SEC_VIO_ADDR	RO	Security violation address for AHB port.	0x0

47.4.52 Security violation address for AHB port 9

This is the most recent security violation address for AHB port 9. However, please refer to SEC_VIO_INFO_VALID register to verify if that slave has valid violation.

Table 939. Security violation address for AHB port 9 (sec_vio_addr9, offset = 0xE24)

Bit	Symbol	Access	Description	Reset value
31:0	SEC_VIO_ADDR	RO	Security violation address for AHB port.	0x0

47.4.53 Security violation address for AHB port 10

This is the most recent security violation address for AHB port 10. However, please refer to SEC_VIO_INFO_VALID register to verify if that slave has valid violation.

Table 940. Security violation address for AHB port 10 (sec_vio_addr10, offset = 0xE28)

Bit	Symbol	Access	Description	Reset value
31:0	SEC_VIO_ADDR	RO	Security violation address for AHB port.	0x0

47.4.54 Security violation address for AHB port 11

This is the most recent security violation address for AHB port 11. However, please refer to SEC_VIO_INFO_VALID register to verify if that slave has valid violation.

Table 941. Security violation address for AHB port 11 (sec_vio_addr11, offset = 0xE2C)

Bit	Symbol	Access	Description	Reset value
31:0	SEC_VIO_ADDR	RO	Security violation address for AHB port.	0x0

47.4.55 Security violation miscellaneous information for AHB port 0

This register provides more details on most recent security violation on AHB port 0.

Table 942. Security violation miscellaneous information for AHB port 0 (sec_vio_misc_info0, offset = 0xE80)

Bit	Symbol	Access	Description	Reset value
0	SEC_VIO_INFO_WRITE	RO	Security violation access read/write indicator, 0: read, 1: write	0x0
1	SEC_VIO_INFO_DATA_ACCESS	RO	Security violation access data/code indicator, 0: code, 1	0x0
3:2		RO	Reserved	undefined
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	RO	Bit [5:4]: master sec level and privileged level bit [7:6]: anti-pol value for master sec level and privileged level	0x0
11:8	SEC_VIO_INFO_MASTER	RO	Security violation master number	0x0
			0: CPU0 Code-bus.	
			1: CPU0 System-bus.	
			2: CPU1 Code-bus.	
			3: CPU1 System-bus.	
			4: USB-FS Device.	
			5: SDMA0.	
			6 - 7: Reserved.	
			8: SDIO	

Table 942. Security violation miscellaneous information for AHB port 0 (sec_vio_misc_info0, offset = 0xE80)

Bit	Symbol	Access	Description	Reset value
			9: PowerQuad (DSP HW Accelerator)	
			10: SHA-2.	
			11: USB-FS Host.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

47.4.56 Security violation miscellaneous information for AHB port 1

This register provides more details on most recent security violation on AHB port 1.

Table 943. Security violation miscellaneous information for AHB port 1 (sec_vio_misc_info1, offset = 0xE84)

Bit	Symbol	Access	Description	Reset value
0	SEC_VIO_INFO_WRITE	RO	Security violation access read/write indicator, 0: read, 1: write	0x0
1	SEC_VIO_INFO_DATA_ACCESS	RO	Security violation access data/code indicator, 0: code, 1	0x0
3:2		RO	Reserved	undefined
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	RO	Bit [5:4]: master sec level and privileged level bit [7:6]: anti-pol value for master sec level and privileged level	0x0
11:8	SEC_VIO_INFO_MASTER	RO	Security violation master number	0x0
			0: CPU0 Code-bus.	
			1: CPU0 System-bus.	
			2: CPU1 Code-bus.	
			3: CPU1 System-bus.	
			4: USB-FS Device.	
			5: SDMA0.	
			6 - 7: Reserved.	
			8: SDIO	
			9: PowerQuad (DSP HW Accelerator)	
			10: SHA-2.	
			11: USB-FS Host.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

47.4.57 Security violation miscellaneous information for AHB port 2

This register provides more details on most recent security violation on AHB port 2.

Table 944. Security violation miscellaneous information for AHB port 2 (sec_vio_misc_info2, offset = 0xE88)

Bit	Symbol	Access	Description	Reset value
0	SEC_VIO_INFO_WRITE	RO	Security violation access read/write indicator, 0: read, 1: write	0x0
1	SEC_VIO_INFO_DATA_ACCESS	RO	Security violation access data/code indicator, 0: code, 1	0x0
3:2		RO	Reserved	undefined
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	RO	bit [5:4]: master sec level and privileged level bit [7:6]: anti-pol value for master sec level and privileged level	0x0
11:8	SEC_VIO_INFO_MASTER	RO	Security violation master number	0x0
			0: CPU0 Code-bus.	
			1: CPU0 System-bus.	
			2: CPU1 Code-bus.	
			3: CPU1 System-bus.	
			4: USB-FS Device.	
			5: SDMA0.	
			6 - 7: Reserved.	
			8: SDIO	
			9: PowerQuad (DSP HW Accelerator)	
			10: SHA-2.	
			11: USB-FS Host.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

47.4.58 Security violation miscellaneous information for AHB port 3

This register provides more details on most recent security violation on AHB port 3.

Table 945. Security violation miscellaneous information for AHB port 3 (sec_vio_misc_info3, offset = 0xE8C)

Bit	Symbol	Access	Description	Reset value
0	SEC_VIO_INFO_WRITE	RO	Security violation access read/write indicator, 0: read, 1: write	0x0
1	SEC_VIO_INFO_DATA_ACCESS	RO	Security violation access data/code indicator, 0: code, 1	0x0
3:2		RO	Reserved	undefined
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	RO	Bit [5:4]: master sec level and privileged level bit [7:6]: anti-pol value for master sec level and privileged level	0x0
11:8	SEC_VIO_INFO_MASTER	RO	Security violation master number	0x0
			0: CPU0 Code-bus.	
			1: CPU0 System-bus.	
			2: CPU1 Code-bus.	
			3: CPU1 System-bus.	

Table 945. Security violation miscellaneous information for AHB port 3 (sec_vio_misc_info3, offset = 0xE8C)

Bit	Symbol	Access	Description	Reset value
			4: USB-FS Device.	
			5: SDMA0.	
			6 - 7: Reserved.	
			8: SDIO	
			9: PowerQuad (DSP HW Accelerator)	
			10: SHA-2.	
			11: USB-FS Host.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

47.4.59 Security violation miscellaneous information for AHB port 4

This register provides more details on most recent security violation on AHB port 4.

Table 946. Security violation miscellaneous information for AHB port 4 (sec_vio_misc_info4, offset = 0xE90)

Bit	Symbol	Access	Description	Reset value
0	SEC_VIO_INFO_WRITE	RO	Security violation access read/write indicator, 0: read, 1: write	0x0
1	SEC_VIO_INFO_DATA_ACCESS	RO	Security violation access data/code indicator, 0: code, 1	0x0
3:2		RO	Reserved	undefined
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	RO	Bit [5:4]: master sec level and privileged level bit [7:6]: anti-pol value for master sec level and privileged level	0x0
11:8	SEC_VIO_INFO_MASTER	RO	Security violation master number	0x0
			0: CPU0 Code-bus.	
			1: CPU0 System-bus.	
			2: CPU1 Code-bus.	
			3: CPU1 System-bus.	
			4: USB-FS Device.	
			5: SDMA0.	
			6 - 7: Reserved.	
			8: SDIO	
			9: PowerQuad (DSP HW Accelerator)	
			10: SHA-2.	
			11: USB-FS Host.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

47.4.60 Security violation miscellaneous information for AHB port 5

This register provides more details on most recent security violation on AHB port 5.

Table 947. Security violation miscellaneous information for AHB port n (sec_vio_misc_info5, offset = 0xE94)

Bit	Symbol	Access	Description	Reset value
0	SEC_VIO_INFO_WRITE	RO	Security violation access read/write indicator, 0: read, 1: write	0x0
1	SEC_VIO_INFO_DATA_ACCESS	RO	Security violation access data/code indicator, 0: code, 1	0x0
3:2		RO	Reserved	undefined
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	RO	Bit [5:4]: master sec level and privileged level bit [7:6]: anti-pol value for master sec level and privileged level	0x0
11:8	SEC_VIO_INFO_MASTER	RO	Security violation master number	0x0
			0: CPU0 Code-bus.	
			1: CPU0 System-bus.	
			2: CPU1 Code-bus.	
			3: CPU1 System-bus.	
			4: USB-FS Device.	
			5: SDMA0.	
			6 - 7: Reserved.	
			8: SDIO	
			9: PowerQuad (DSP HW Accelerator)	
			10: SHA-2.	
			11: USB-FS Host.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

47.4.61 Security violation miscellaneous information for AHB port 6

This register provides more details on most recent security violation on AHB port 6.

Table 948. Security violation miscellaneous information for AHB port 6 (sec_vio_misc_info6, offset = 0xE98)

Bit	Symbol	Access	Description	Reset value
0	SEC_VIO_INFO_WRITE	RO	Security violation access read/write indicator, 0: read, 1: write	0x0
1	SEC_VIO_INFO_DATA_ACCESS	RO	Security violation access data/code indicator, 0: code, 1	0x0
3:2		RO	Reserved	undefined
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	RO	Bit [5:4]: master sec level and privileged level bit [7:6]: anti-pol value for master sec level and privileged level	0x0
11:8	SEC_VIO_INFO_MASTER	RO	Security violation master number	0x0
			0: CPU0 Code-bus.	
			1: CPU0 System-bus.	
			2: CPU1 Code-bus.	
			3: CPU1 System-bus.	

Table 948. Security violation miscellaneous information for AHB port 6 (sec_vio_misc_info6, offset = 0xE98)

Bit	Symbol	Access	Description	Reset value
			4: USB-FS Device.	
			5: SDMA0.	
			6 - 7: Reserved.	
			8: SDIO	
			9: PowerQuad (DSP HW Accelerator)	
			10: SHA-2.	
			11: USB-FS Host.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

47.4.62 Security violation miscellaneous information for AHB port 7

This register provides more details on most recent security violation on AHB port 7.

Table 949. Security violation miscellaneous information for AHB port 7 (sec_vio_misc_info6, offset = 0xE9C)

Bit	Symbol	Access	Description	Reset value
0	SEC_VIO_INFO_WRITE	RO	Security violation access read/write indicator, 0: read, 1: write	0x0
1	SEC_VIO_INFO_DATA_ACCESS	RO	Security violation access data/code indicator, 0: code, 1	0x0
3:2		RO	Reserved	undefined
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	RO	Bit [5:4]: master sec level and privileged level bit [7:6]: anti-pol value for master sec level and privileged level	0x0
11:8	SEC_VIO_INFO_MASTER	RO	Security violation master number	0x0
			0: CPU0 Code-bus.	
			1: CPU0 System-bus.	
			2: CPU1 Code-bus.	
			3: CPU1 System-bus.	
			4: USB-FS Device.	
			5: SDMA0.	
			6 - 7: Reserved.	
			8: SDIO	
			9: PowerQuad (DSP HW Accelerator)	
			10: SHA-2.	
			11: USB-FS Host.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

47.4.63 Security violation miscellaneous information for AHB port 8

This register provides more details on most recent security violation on AHB port 8.

Table 950. Security violation miscellaneous information for AHB port 8 (sec_vio_misc_info7, offset = 0xEA0)

Bit	Symbol	Access	Description	Reset value
0	SEC_VIO_INFO_WRITE	RO	Security violation access read/write indicator, 0: read, 1: write	0x0
1	SEC_VIO_INFO_DATA_ACCESS	RO	Security violation access data/code indicator, 0: code, 1	0x0
3:2		RO	Reserved	undefined
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	RO	Bit [5:4]: master sec level and privileged level bit [7:6]: anti-pol value for master sec level and privileged level	0x0
11:8	SEC_VIO_INFO_MASTER	RO	Security violation master number	0x0
			0: CPU0 Code-bus.	
			1: CPU0 System-bus.	
			2: CPU1 Code-bus.	
			3: CPU1 System-bus.	
			4: USB-FS Device.	
			5: SDMA0.	
			6 - 7: Reserved.	
			8: SDIO	
			9: PowerQuad (DSP HW Accelerator)	
			10: SHA-2.	
			11: USB-FS Host.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

47.4.64 Security violation miscellaneous information for AHB port 9

This register provides more details on most recent security violation on AHB port 9.

Table 951. Security violation miscellaneous information for AHB port 9 (sec_vio_misc_info8, offset = 0xEA4)

Bit	Symbol	Access	Description	Reset value
0	SEC_VIO_INFO_WRITE	RO	Security violation access read/write indicator, 0: read, 1: write	0x0
1	SEC_VIO_INFO_DATA_ACCESS	RO	Security violation access data/code indicator, 0: code, 1	0x0
3:2		RO	Reserved	undefined
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	RO	Bit [5:4]: master sec level and privileged level bit [7:6]: anti-pol value for master sec level and privileged level	0x0
11:8	SEC_VIO_INFO_MASTER	RO	Security violation master number	0x0
			0: CPU0 Code-bus.	
			1: CPU0 System-bus.	
			2: CPU1 Code-bus.	
			3: CPU1 System-bus.	

Table 951. Security violation miscellaneous information for AHB port 9 (sec_vio_misc_info8, offset = 0xEA4)

Bit	Symbol	Access	Description	Reset value
			4: USB-FS Device.	
			5: SDMA0.	
			6 - 7: Reserved.	
			8: SDIO	
			9: PowerQuad (DSP HW Accelerator)	
			10: SHA-2.	
			11: USB-FS Host.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

47.4.65 Security violation miscellaneous information for AHB port 10

This register provides more details on most recent security violation on AHB port 10.

Table 952. Security violation miscellaneous information for AHB port 10 (sec_vio_misc_info10, offset = 0xEA8)

Bit	Symbol	Access	Description	Reset value
0	SEC_VIO_INFO_WRITE	RO	Security violation access read/write indicator, 0: read, 1: write	0x0
1	SEC_VIO_INFO_DATA_ACCESS	RO	Security violation access data/code indicator, 0: code, 1	0x0
3:2		RO	Reserved	undefined
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	RO	Bit [5:4]: master sec level and privileged level bit [7:6]: anti-pol value for master sec level and privileged level	0x0
11:8	SEC_VIO_INFO_MASTER	RO	Security violation master number	0x0
			0: CPU0 Code-bus.	
			1: CPU0 System-bus.	
			2: CPU1 Code-bus.	
			3: CPU1 System-bus.	
			4: USB-FS Device.	
			5: SDMA0.	
			6 - 7: Reserved.	
			8: SDIO	
			9: PowerQuad (DSP HW Accelerator)	
			10: SHA-2.	
			11: USB-FS Host.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

47.4.66 Security violation miscellaneous information for AHB port 11

This register provides more details on most recent security violation on AHB port 11.

Table 953. Security violation miscellaneous information for AHB port 11 (sec_vio_misc_info11, offset = 0xEAC)

Bit	Symbol	Access	Description	Reset value
0	SEC_VIO_INFO_WRITE	RO	Security violation access read/write indicator, 0: read, 1: write	0x0
1	SEC_VIO_INFO_DATA_ACCESS	RO	Security violation access data/code indicator, 0: code, 1	0x0
3:2		RO	Reserved	undefined
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	RO	Bit [5:4]: master sec level and privileged level bit [7:6]: anti-pol value for master sec level and privileged level	0x0
11:8	SEC_VIO_INFO_MASTER	RO	Security violation master number	0x0
			0: CPU0 Code-bus.	
			1: CPU0 System-bus.	
			2: CPU1 Code-bus.	
			3: CPU1 System-bus.	
			4: USB-FS Device.	
			5: SDMA0.	
			6 - 7: Reserved.	
			8: SDIO	
			9: PowerQuad (DSP HW Accelerator)	
			10: SHA-2.	
			11: USB-FS Host.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

47.4.67 Security violation address/information registers valid flags

This register describes if security violation happened on a given slave. If valid=1, look at vio_addr0 - sec_vio_addrx and sec_vio_misc_infox registers.

Table 954. Security violation address/information registers valid flags (SEC_VIO_INFO_VALID, offset = 0xF00)

BIt	Symbol	Description	Reset value
0	VIO_INFO_VALID0	Violation information valid flag for AHB port 0. 0: not valid. 1: valid (violation occurred). WrIte 1 to clear.	0x0
1	VIO_INFO_VALID1	Violation information valid flag for AHB port 1. 0: not valid. 1: valid (violation occurred). WrIte 1 to clear.	0x0
2	VIO_INFO_VALID2	Violation information valid flag for AHB port 2. 0: not valid. 1: valid (violation occurred). WrIte 1 to clear.	0x0
3	VIO_INFO_VALID3	Violation information valid flag for AHB port 3. 0: not valid. 1: valid (violation occurred). WrIte 1 to clear.	0x0
4	VIO_INFO_VALID4	Violation information valid flag for AHB port 4. 0: not valid. 1: valid (violation occurred). WrIte 1 to clear.	0x0
5	VIO_INFO_VALID5	Violation information valid flag for AHB port 5. 0: not valid. 1: valid (violation occurred). WrIte 1 to clear.	0x0

Table 954. Security violation address/information registers valid flags (SEC_VIO_INFO_VALID, offset = 0xF00)

Bit	Symbol	Description	Reset value
6	VIO_INFO_VALID6	Violation information valid flag for AHB port 6. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
7	VIO_INFO_VALID7	Violation information valid flag for AHB port 7. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
8	VIO_INFO_VALID8	Violation information valid flag for AHB port 8. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
9	VIO_INFO_VALID9	Violation information valid flag for AHB port 9. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
10	VIO_INFO_VALID10	Violation information valid flag for AHB port 10. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
11	VIO_INFO_VALID11	Violation information valid flag for AHB port 11. 0: not valid. 1: valid (violation occurred). Write 1 to clear.	0x0
31:12	-	Reserved.	undefined

47.4.68 Secure GPIO mask for port 0 pins

This register is used to block leakage of Secure interface (GPIOs, I2C, UART, and other peripherals configured as Secure peripherals) pin states to Non-secure world.

If this port is not masked, its port pin states can be read using normal GPIO port even if these port pins are configured as other digital functions (UART, I2C) than GPIO. If masked, GPIO IP would read the state as 0 independent of the activity on the Pin.

This register controls masking for Port0 pins.

Table 955. Secure GPIO mask for port 0 pins (SEC_GPIO_MASK0, offset = 0xF80)

Bit	Symbol	Description	Reset value
0	PIO0_PIN0_SEC_MASK	0 = GPIO can read value from IO P0(0) 1= GPIO cannot read value from IO P0(0)	0x1
1	PIO0_PIN1_SEC_MASK	0 = GPIO can read value from IO P0(1) 1= GPIO cannot read value from IO P0(1)	0x1
2	PIO0_PIN2_SEC_MASK	0 = GPIO can read value from IO P0(2) 1= GPIO cannot read value from IO P0(2)	0x1
3	PIO0_PIN3_SEC_MASK	0 = GPIO can read value from IO P0(3) 1= GPIO cannot read value from IO P0(3)	0x1
4	PIO0_PIN4_SEC_MASK	0 = GPIO can read value from IO P0(4) 1= GPIO cannot read value from IO P0(4)	0x1
5	PIO0_PIN5_SEC_MASK	0 = GPIO can read value from IO P0(5) 1= GPIO cannot read value from IO P0(5)	0x1
6	PIO0_PIN6_SEC_MASK	0 = GPIO can read value from IO P0(6) 1= GPIO cannot read value from IO P0(6)	0x1
7	PIO0_PIN7_SEC_MASK	0 = GPIO can read value from IO P0(7) 1= GPIO cannot read value from IO P0(7)	0x1
8	PIO0_PIN8_SEC_MASK	0 = GPIO can read value from IO P0(8) 1= GPIO cannot read value from IO P0(8)	0x1

Table 955. Secure GPIO mask for port 0 pins (SEC_GPIO_MASK0, offset = 0xF80) *…continued*

Bit	Symbol	Description	Reset value
9	PIO0_PIN9_SEC_MASK	0 = GPIO can read value from IO P0(9) 1= GPIO cannot read value from IO P0(9)	0x1
10	PIO0_PIN10_SEC_MASK	0 = GPIO can read value from IO P0(10) 1= GPIO cannot read value from IO P0(10)	0x1
11	PIO0_PIN11_SEC_MASK	0 = GPIO can read value from IO P0(11) 1= GPIO cannot read value from IO P0(11)	0x1
12	PIO0_PIN12_SEC_MASK	0 = GPIO can read value from IO P0(12) 1= GPIO cannot read value from IO P0(12)	0x1
13	PIO0_PIN13_SEC_MASK	0 = GPIO can read value from IO P0(13) 1= GPIO cannot read value from IO P0(13)	0x1
14	PIO0_PIN14_SEC_MASK	0 = GPIO can read value from IO P0(14) 1= GPIO cannot read value from IO P0(14)	0x1
15	PIO0_PIN15_SEC_MASK	0 = GPIO can read value from IO P0(15) 1= GPIO cannot read value from IO P0(15)	0x1
16	PIO0_PIN16_SEC_MASK	0 = GPIO can read value from IO P0(16) 1= GPIO cannot read value from IO P0(16)	0x1
17	PIO0_PIN17_SEC_MASK	0 = GPIO can read value from IO P0(17) 1= GPIO cannot read value from IO P0(17)	0x1
18	PIO0_PIN18_SEC_MASK	0 = GPIO can read value from IO P0(18) 1= GPIO cannot read value from IO P0(18)	0x1
19	PIO0_PIN19_SEC_MASK	0 = GPIO can read value from IO P0(19) 1= GPIO cannot read value from IO P0(19)	0x1
20	PIO0_PIN20_SEC_MASK	0 = GPIO can read value from IO P0(20) 1= GPIO cannot read value from IO P0(20)	0x1
21	PIO0_PIN21_SEC_MASK	0 = GPIO can read value from IO P0(21) 1= GPIO cannot read value from IO P0(21)	0x1
22	PIO0_PIN22_SEC_MASK	0 = GPIO can read value from IO P0(22) 1= GPIO cannot read value from IO P0(22)	0x1
23	PIO0_PIN23_SEC_MASK	0 = GPIO can read value from IO P0(23) 1= GPIO cannot read value from IO P0(23)	0x1
24	PIO0_PIN24_SEC_MASK	0 = GPIO can read value from IO P0(24) 1= GPIO cannot read value from IO P0(24)	0x1
25	PIO0_PIN25_SEC_MASK	0 = GPIO can read value from IO P0(25) 1= GPIO cannot read value from IO P0(25)	0x1
26	PIO0_PIN26_SEC_MASK	0 = GPIO can read value from IO P0(26) 1= GPIO cannot read value from IO P0(26)	0x1
27	PIO0_PIN27_SEC_MASK	0 = GPIO can read value from IO P0(27) 1= GPIO cannot read value from IO P0(27)	0x1
28	PIO0_PIN28_SEC_MASK	0 = GPIO can read value from IO P0(28) 1= GPIO cannot read value from IO P0(28)	0x1

Table 955. Secure GPIO mask for port 0 pins (SEC_GPIO_MASK0, offset = 0xF80) *…continued*

Bit	Symbol	Description	Reset value
29	PIO0_PIN29_SEC_MASK	0 = GPIO can read value from IO P0(29) 1= GPIO cannot read value from IO P0(29)	0x1
30	PIO0_PIN30_SEC_MASK	0 = GPIO can read value from IO P0(30) 1= GPIO cannot read value from IO P0(30)	0x1
31	PIO0_PIN31_SEC_MASK	0 = GPIO can read value from IO P0(31) 1= GPIO cannot read value from IO P0(31)	0x1

47.4.69 Secure GPIO mask for port 1 pins

This register is used to block leakage of Secure interface (GPIOs, I2C, UART, and other peripherals configured as Secure peripherals) pin states to Non-secure world.

If this port is not masked, its port pin states can be read using normal GPIO port even if these port pins are configured as other digital functions (UART, I2C) than GPIO. If masked, GPIO IP would read the state as 0 independent of the activity on the Pin.

This register controls masking for Port1 pins.

Table 956. Secure GPIO mask for port 1 pins (SEC_GPIO_MASK1, offset = 0xF84)

Bit	Symbol	Description	Reset value
0	PIO1_PIN0_SEC_MASK	0 = GPIO can read value from IO P1(0) 1= GPIO cannot read value from IO P1(0)	0x1
1	PIO1_PIN1_SEC_MASK	0 = GPIO can read value from IO P1(1) 1= GPIO cannot read value from IO P1(1)	0x1
2	PIO1_PIN2_SEC_MASK	0 = GPIO can read value from IO P1(2) 1= GPIO cannot read value from IO P1(2)	0x1
3	PIO1_PIN3_SEC_MASK	0 = GPIO can read value from IO P1(3) 1= GPIO cannot read value from IO P1(3)	0x1
4	PIO1_PIN4_SEC_MASK	0 = GPIO can read value from IO P1(4) 1= GPIO cannot read value from IO P1(4)	0x1
5	PIO1_PIN5_SEC_MASK	0 = GPIO can read value from IO P1(5) 1= GPIO cannot read value from IO P1(5)	0x1
6	PIO1_PIN6_SEC_MASK	0 = GPIO can read value from IO P1(6) 1= GPIO cannot read value from IO P1(6)	0x1
7	PIO1_PIN7_SEC_MASK	0 = GPIO can read value from IO P1(7) 1= GPIO cannot read value from IO P1(7)	0x1
8	PIO1_PIN8_SEC_MASK	0 = GPIO can read value from IO P1(8) 1= GPIO cannot read value from IO P1(8)	0x1
9	PIO1_PIN9_SEC_MASK	0 = GPIO can read value from IO P1(9) 1= GPIO cannot read value from IO P1(9)	0x1
10	PIO1_PIN10_SEC_MASK	0 = GPIO can read value from IO P1(10) 1= GPIO cannot read value from IO P1(10)	0x1
11	PIO1_PIN11_SEC_MASK	0 = GPIO can read value from IO P1(11) 1= GPIO cannot read value from IO P1(11)	0x1

Table 956. Secure GPIO mask for port 1 pins (SEC_GPIO_MASK1, offset = 0xF84) *…continued*

Bit	Symbol	Description	Reset value
12	PIO1_PIN12_SEC_MASK	0 = GPIO can read value from IO P1(12) 1= GPIO cannot read value from IO P1(12)	0x1
13	PIO1_PIN13_SEC_MASK	0 = GPIO can read value from IO P1(13) 1= GPIO cannot read value from IO P1(13)	0x1
14	PIO1_PIN14_SEC_MASK	0 = GPIO can read value from IO P1(14) 1= GPIO cannot read value from IO P1(14)	0x1
15	PIO1_PIN15_SEC_MASK	0 = GPIO can read value from IO P1(15) 1= GPIO cannot read value from IO P1(15)	0x1
16	PIO1_PIN16_SEC_MASK	0 = GPIO can read value from IO P1(16) 1= GPIO cannot read value from IO P1(16)	0x1
17	PIO1_PIN17_SEC_MASK	0 = GPIO can read value from IO P1(17) 1= GPIO cannot read value from IO P1(17)	0x1
18	PIO1_PIN18_SEC_MASK	0 = GPIO can read value from IO P1(18) 1= GPIO cannot read value from IO P1(18)	0x1
19	PIO1_PIN19_SEC_MASK	0 = GPIO can read value from IO P1(19) 1= GPIO cannot read value from IO P1(19)	0x1
20	PIO1_PIN20_SEC_MASK	0 = GPIO can read value from IO P1(20) 1= GPIO cannot read value from IO P1(20)	0x1
21	PIO1_PIN21_SEC_MASK	0 = GPIO can read value from IO P1(21) 1= GPIO cannot read value from IO P1(21)	0x1
22	PIO1_PIN22_SEC_MASK	0 = GPIO can read value from IO P1(22) 1= GPIO cannot read value from IO P1(22)	0x1
23	PIO1_PIN23_SEC_MASK	0 = GPIO can read value from IO P1(23) 1= GPIO cannot read value from IO P1(23)	0x1
24	PIO1_PIN24_SEC_MASK	0 = GPIO can read value from IO P1(24) 1= GPIO cannot read value from IO P1(24)	0x1
25	PIO1_PIN25_SEC_MASK	0 = GPIO can read value from IO P1(25) 1= GPIO cannot read value from IO P1(25)	0x1
26	PIO1_PIN26_SEC_MASK	0 = GPIO can read value from IO P1(26) 1= GPIO cannot read value from IO P1(26)	0x1
27	PIO1_PIN27_SEC_MASK	0 = GPIO can read value from IO P1(27) 1= GPIO cannot read value from IO P1(27)	0x1
28	PIO1_PIN28_SEC_MASK	0 = GPIO can read value from IO P1(28) 1= GPIO cannot read value from IO P1(28)	0x1
29	PIO1_PIN29_SEC_MASK	0 = GPIO can read value from IO P1(29) 1= GPIO cannot read value from IO P1(29)	0x1
30	PIO1_PIN30_SEC_MASK	0 = GPIO can read value from IO P1(30) 1= GPIO cannot read value from IO P1(30)	0x1
31	PIO1_PIN31_SEC_MASK	0 = GPIO can read value from IO P1(31) 1= GPIO cannot read value from IO P1(31)	0x1

47.4.70 Secure interrupt mask for CPU1

This register can block different peripheral interrupt from IRQ bus to go to CPU1. If CPU1 is not a secure master, application might want to disable secure peripheral interrupts reaching that CPU

Table 957. Secure interrupt mask for CPU1 (SEC_CPU_INT_MASK0, offset = 0xF90)

Bit	Symbol	Value	Description	Reset value
0	SYS_IRQ		Watchdog Timer, Brown Out Detectors and Flash Controller interrupts	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
1	SDMA0_IRQ		System DMA 0 (Non-secure) interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
2	GPIO_GLOBALINT0_IRQ		GPIO group 0 interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
3	GPIO_GLOBALINT1_IRQ		GPIO group 1 interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
4	GPIO_INT0_IRQ0		Pin interrupt 0 or pattern match engine slice 0 interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
5	GPIO_INT0_IRQ1		Pin interrupt 1 or pattern match engine slice 1 interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
6	GPIO_INT0_IRQ2		Pin interrupt 2 or pattern match engine slice 2 interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
7	GPIO_INT0_IRQ3		Pin interrupt 3 or pattern match engine slice 3 interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
8	UTICK_IRQ		Micro Tick timer interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
9	MRT_IRQ		Multi-Rate timer interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
10	CTIMER0_IRQ		Standard counter/timer 0 interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
11	CTIMER1_IRQ		Standard counter/timer 1 interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	

Table 957. Secure interrupt mask for CPU1 (SEC_CPU_INT_MASK0, offset = 0xF90) *...continued*

Bit	Symbol	Value	Description	Reset value
12	SCT_IRQ		SCTimer/PWM interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
13	CTIMER3_IRQ		Standard counter/timer 3 interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
14	FLEXCOMM0_IRQ		Flexcomm interface 0 interrupt (USART, SPI, I^2C, I^2S).	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
15	FLEXCOMM1_IRQ		Flexcomm interface 1 interrupt (USART, SPI, I^2C, I^2S).	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
17:16	FLEXCOMM2_IRQ		Flexcomm interface 2 interrupt (USART, SPI, I^2C, I^2S).	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
	FLEXCOMM3_IRQ		Flexcomm interface 3 interrupt (USART, SPI, I^2C, I^2S).	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
18	FLEXCOMM4_IRQ		Flexcomm interface 4 interrupt (USART, SPI, I^2C, I^2S).	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
19	FLEXCOMM5_IRQ		Flexcomm interface 5 interrupt (USART, SPI, I^2C, I^2S).	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
20	FLEXCOMM6_IRQ		Flexcomm interface 6 interrupt (USART, SPI, I^2C, I^2S).	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
21	FLEXCOMM7_IRQ		Flexcomm interface interfdace7 interrupt (USART, SPI, I^2C, I^2S).	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
22	ADC_IRQ		General purpose ADC interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
23	RESERVED 0		Reserved. Read value is undefined, only zero should be wrltten.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
24	ACMP_IRQ		Analog comparator interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	

Table 957. Secure interrupt mask for CPU1 (SEC_CPU_INT_MASK0, offset = 0xF90) *...continued*

Bit	Symbol	Value	Description	Reset value
25	RESERVED 1		Reserved. Read value is undefined, only zero should be wrltten.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
26	Reserved 2		Reserved. Read value is undefined, only zero should be wrltten.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
27	USB0_NEEDCLK		USB full speed controller clock request interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
28	USB0_IRQ		USB high speed controller interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
29	RTC_IRQ		RTC_LITE0_ALARM_IRQ, RTC_LITE0_WAKEUP_IRQ	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	
31	MAILBOX_IRQ		Mailbox interrupt.	0x1
		0	0 = Interrupt is not visible to CPU1	
		1	1 = Interrupt is visible to CPU1	

47.4.71 Secure interrupt mask for CPU1

This register can block different peripheral interrupt from IRQ bus to go to CPU1. If CPU1 is not a secure master, application might want to disable secure peripheral interrupts reaching that CPU

Table 958. Secure interrupt mask for CPU1 (SEC_CPU_INT_MASK1, offset = 0xF94)

Bit	Symbol	Value	Description	Reset value
0	GPIO_INT0_IRQ4		Pin interrupt 4 or pattern match engine slice 4 interrupt.	0x1
		0		
		1		
1	GPIO_INT0_IRQ5		Pin interrupt 5 or pattern match engine slice 5 interrupt.	0x1
		0		
		1		
2	GPIO_INT0_IRQ6		Pin interrupt 6 or pattern match engine slice 6 interrupt.	0x1
		0		
		1		
3	GPIO_INT0_IRQ7		Pin interrupt 7 or pattern match engine slice 7 interrupt.	0x1
		0		
		1		
4	CTIMER2_IRQ		Standard counter/timer 2 interrupt.	0x1
		0		
		1		

Table 958. Secure interrupt mask for CPU1 (SEC_CPU_INT_MASK1, offset = 0xF94) *…continued*

Bit	Symbol	Value	Description	Reset value
5	CTIMER4_IRQ		Standard counter/timer 4 interrupt.	0x1
		0		
		1		
6	OS_EVENT_TIMER_IRQ		OS event timer and OS event timer wake up interrupts	0x1
		0		
		1		
7	RESERVED 0		Reserved. Read value is undefined, only zero should be written.	0x1
		0		
		1		
8	RESERVED 1		Reserved. Read value is undefined, only zero should be written.	0x1
		0		
		1		
9	RESERVED 2		Reserved. Read value is undefined, only zero should be written.	0x1
		0		
		1		
10	SDIO_IRQ		SDIO controller interrupt.	0x1
		0		
		1		
11	RESERVED 3		Reserved. Read value is undefined, only zero should be written.	0x1
		0		
		1		
12	RESERVED 4		Reserved. Read value is undefined, only zero should be written.	0x1
		0		
		1		
13	RESERVED 5		Reserved. Read value is undefined, only zero should be written.	0x1
		0		
		1		
14	USB1_PHY		USB high speed controller PHY interrupt.	0x1
		0		
		1		
15	USB1_IRQ		USB high speed controller interrupt.	0x1
		0		
		1		
16	USB1_NEEDCLK		USB high speed controller clock request interrupt.	0x1
		0		
		1		
17	SEC_HYPERVISOR_CALL_IRQ		Secure fault hyper visor call interrupt.	0x1
		0		
		1		

Table 958. Secure interrupt mask for CPU1 (SEC_CPU_INT_MASK1, offset = 0xF94) *…continued*

Bit	Symbol	Value	Description	Reset value
18	SEC_GPIO_INT0_IRQ0		Secure pin interrupt 0 or pattern match engine slice 0 interrupt.	0x1
		0		
		1		
19	SEC_GPIO_INT0_IRQ1		Secure pin interrupt 1 or pattern match engine slice 1 interrupt.	0x1
		0		
		1		
20	PLU_IRQ		Programmable look-up controller interrupt.	0x1
		0		
		1		
21	SEC_VIO_IRQ		Security violation interrupt.	0x1
		0		
		1		
22	SHA_IRQ		HASH-AES interrupt.	0x1
		0		
		1		
23	CASPER_IRQ		CASPER interrupt.	0x1
		0		
		1		
24	PUFKEY_IRQ		PUF interrupt.	0x1
		0		
		1		
25	PQ_IRQ		Power quad interrupt.	0x1
		0		
		1		
26	SDMA1_IRQ		System DMA 1 (Secure) interrupt.	0x1
		0		
		1		
27	LSPI_HS_IRQ		High speed SPI interrupt.	0x1
		0		
		1		
31:28	-	-	Reserved.	undefined

47.4.72 Security general purpose register access control

This register allows locking of other registers. Each field are individually writable. Once written with the lock value, they can be reverted only by system reset.

Table 959. Security general purpose register access control. (SEC_MASK_LOCK, offset = 0xFBC)

Bit	Symbol	Value	Description	Reset value
1:0	SEC_GPIO_MASK0_LOCK		Secure GPIO _MASK0 lock.	0x2
		0x2	SEC_GPIO_MASK0 can be wrltten.	
		0x1	SEC_REG_REG0 cannot be wrltten.	
		0x0		
		0x3		
3:2	SEC_GPIO_MASK1_LOCK		Secure GPIO MASK1 lock.	0x2
		0x2	SEC_GPIO_MASK1 can be wrltten.	
		0x1	SEC_REG_REG1 cannot be wrltten.	
		0x0		
		0x3		
7:4	-		Reserved.	
9:8	SEC_CPU1_INT_MASK0_LOCK		SEC_CPU1_INT_MASK0 lock.	0x2
		0x2	SEC_CPU1_INT_MASK0 can be written.	
		0x1	SEC_REG_REG4 cannot be wrltten.	
		0x0		
		0x3		
11:10	SEC_CPU1_INT_MASK1_LOCK		SEC_CPU1_INT_MASK1 lock.	0x2
		0x2	SEC_CPU1_INT_MASK1 can be written.	
		0x1	SEC_REG_REG5 cannot be wrltten.	
		0x0		
		0x3		
31:12	-	-	Reserved.	undefined

47.4.73 Master secure level register

This register allows configuring security level for each master on AHB. Expectation is that application makes a static choice up front; programs and locks this register with the help of ROM. Once LOCK (bit 31-30) is applied, it can be unlocked only by system reset.

Table 960. Master secure level register (MASTER_SEC_LEVEL, offset = 0xFD0)

Bit	Symbol	Value	Description	Reset value
3:0	-		Reserved.	0x0
5:4	MCM33C		Secondary CPU (CPU1) code bus.	0x0
			0x3=Secure privileged 0x2=Secure non-privileged 0x1=non-Secure privileged 0x0=non-Secure non-privileged	

Table 960. Master secure level register (MASTER_SEC_LEVEL, offset = 0xFD0) *…continued*

Bit	Symbol	Value	Description	Reset value
7:6	MCM33S		Secondary CPU (CPU1) system bus.	0x0
			0x3=Secure privileged 0x2=Secure non-privileged 0x1=non-Secure privileged 0x0=non-Secure non-privileged	
9:8	USBFSD		USB full speed Device.	0x0
			0x3=Secure privileged 0x2=Secure non-privileged 0x1=non-Secure privileged 0x0=non-Secure non-privileged	
11:10	SDMA0		System DMA 0.	0x0
			0x3=Secure privileged 0x2=Secure non-privileged 0x1=non-Secure privileged 0x0=non-Secure non-privileged	
17:16	SDIO		SDIO.	0x0
			0x3=Secure privileged 0x2=Secure non-privileged 0x1=non-Secure privileged 0x0=non-Secure non-privileged	
19:18	PQ		Power quad.	0x0
			0x3=Secure privileged 0x2=Secure non-privileged 0x1=non-Secure privileged 0x0=non-Secure non-privileged	
21:20	HASH		Hash.	0x0
			0x3=Secure privileged 0x2=Secure non-privileged 0x1=non-Secure privileged 0x0=non-Secure non-privileged	
23:22	USBFSH		USB full speed host.	0x0
			0x3=Secure privileged 0x2=Secure non-privileged 0x1=non-Secure privileged 0x0=non-Secure non-privileged	
25:24	SDMA1		System DMA 1 security level.	0x0
			0x3=Secure privileged 0x2=Secure non-privileged 0x1=non-Secure privileged 0x0=non-Secure non-privileged	
29:26	-		Reserved. Read value is undefined, only zero should be wrltten.	0x0

Table 960. Master secure level register (MASTER_SEC_LEVEL, offset = 0xFD0) *…continued*

Bit	Symbol	Value	Description	Reset value
31:30	MASTER_SEC_LEVEL_LOCK		MASTER_SEC_LEVEL lock.	0x2
		0x2	MASTER_SEC_LEVEL_LOCK can be wrltten.	
		0x1	MASTER_SEC_LEVEL_LOCK cannot be wrltten.	
		0x0		
		0x3		

47.4.74 Master secure level anti-pole register

This register is inverse of MASTER_SEC_LEVEL register above. Secondary register with inverted programing is implemented to provide better protection against malicious hacking attacks such as glitch attack

Table 961. Master secure level anti-pole register (MASTER_SEC_ANTI_POL_REG, offset = 0xFD4)

Bit	Symbol	Value	Description	Reset value
3:0	-			0xF
5:4	MCM33C		CPU1 (CM33) code bus. Must be equal to NOT(MASTER_SEC_LEVEL.MCM33C).	0x3
			0x0=Secure privileged 0x1=Secure non-privileged 0x2=non-Secure privileged	
7:6	MCM33S		CPU1 (CM33) system bus. Must be equal to NOT(MASTER_SEC_LEVEL.MCM33S).	0x3
			0x0=Secure privileged 0x1=Secure non-privileged 0x2=non-Secure privileged	
9:8	USBFSD		USB full speed device. Must be equal to NOT(MASTER_SEC_LEVEL.USBFSD).	0x3
			0x0=Secure privileged 0x1=Secure non-privileged 0x2=non-Secure privileged	
11:10	SDMA0		System DMA 0. Must be equal to NOT(MASTER_SEC_LEVEL.SDMA0).	0x3
			0x0=Secure privileged 0x1=Secure non-privileged 0x2=non-Secure privileged	
13:12	RESERVED		Reserved. Read value is undefined, only zero should be wrltten.	0x3
15:14	RESERVED		Reserved. Read value is undefined, only zero should be wrltten.	0x3
17:16	SDIO		SDIO. Must be equal to NOT(MASTER_SEC_LEVEL.SDIO).	0x3
			0x0=Secure privileged 0x1=Secure non-privileged 0x2=non-Secure privileged	

Table 961. Master secure level anti-pole register (MASTER_SEC_ANTI_POL_REG, offset = 0xFD4) *…continued*

Bit	Symbol	Value	Description	Reset value
19:18	PQ		Power quad. Must be equal to NOT(MASTER_SEC_LEVEL.PQ)	0x3
			0x0=Secure privileged 0x1=Secure non-privileged 0x2=non-Secure privileged	
21:20	HASH		Hash. Must be equal to NOT(MASTER_SEC_LEVEL.HASH)	0x3
			0x0=Secure privileged 0x1=Secure non-privileged 0x2=non-Secure privileged	
23:22	USBFSH		USB full speed host. Must be equal to NOT(MASTER_SEC_LEVEL.USBFSH)	0x3
			0x0=Secure privileged 0x1=Secure non-privileged 0x2=non-Secure privileged	
25:24	SDMA1		System DMA 1 security level. Must be equal to NOT(MASTER_SEC_LEVEL.SDMA1)	0x3
			0x0=Secure privileged 0x1=Secure non-privileged 0x2=non-Secure privileged	
29:26	-		Reserved. Read value is undefined, only zero should be wrltten.	0xF
31:30	MASTER_SEC_LEVEL_ANTIPOL_LOCK		MASTER_SEC_LEVEL_ANTIPOL lock.	0x2
		0x2	MASTER_SEC_LEVEL_ANTIPOL_LOCK can be written.	
		0x1	MASTER_SEC_LEVEL_ANTIPOL_LOCK cannot be written.	
		0x0		
		0x3		

47.4.75 Miscellaneous control signals for in Primary CPU0

This register drives certain input ports of CPU0, providing capability to lock the settings or enhanced security.

Table 962. Miscellaneous control signals for in CPU0 (CPU0_LOCK_REG, offset = 0xFEC)

Bit	Symbol	Value	Description	Reset value
1:0	LOCK_NS_VTOR		CM33 (CPU0) VTOR_NS register write-lock.	0x2
		0x2	CPU0 LOCKNSVTOR is 0. Enable writes to the VTOR_NS register	
		0x1	CPU0 LOCKNSVTOR is 1. Disable writes to the VTOR_NS register	
		0x0		
		0x3		

Table 962. Miscellaneous control signals for in CPU0 (CPU0_LOCK_REG, offset = 0xFEC) *...continued*

Bit	Symbol	Value	Description	Reset value
3:2	LOCK_NS_MPU		CPU0 non-secure MPU register write-lock	0x2
		0x2	CPU0 LOCKNSMPU is 0. Enable writes to non-secure MPU registers	
		0x1	CPU0 LOCKNSMPU is 1. Disable writes to non-secure MPU registers	
		0x0		
		0x3		
5:4	LOCK_S_VTAIRCR		CPU0 VTOR_S, AIRCR.PRIS, IRCR.BFHFNMINS registers write-lock	0x2
		0x2	CPU0 LOCKSVTAIRCR is 0. Enable writes to the VTOR_S, AIRCR.PRIS, IRCR.BFHFNMINS register	
		0x1	CPU0 LOCKSVTAIRCR is 1. Disable writes to the VTOR_S, AIRCR.PRIS, IRCR.BFHFNMINS registers	
		0x0		
		0x3		
7:6	LOCK_S_MPU		CPU0 Secure MPU registers write-lock	0x2
		0x2	CPU0 LOCKSMPU is 0. Enable writes to secure MPU registers	
		0x1	CPU0 LOCKSMPU is 1. Disable writes to secure MPU registers	
		0x0		
		0x3		
9:8	LOCK_SAU		CPU0 SAU registers write-lock	0x2
		0x2	CPU0 LOCKSAU is 0. Enable writes to SAU_CTRL, SAU_RNR, SAU_RBAR and SAU_RLAR registers	
		0x1	CPU0 LOCKSAU is 1. Disable writes to SAU_CTRL, SAU_RNR, SAU_RBAR and SAU_RLAR registers	
		0x0		
		0x3		
29:10	-		Reserved.	undefined
31:30	CPU0_LOCK_REG_LOCK		Disables write access to this register itself. Once locked, only system reset can unlock (to enable write access)	0x2
		0x2	This register can be written.	
		0x1	This register can't be written (included this bitfield)	
		0x0		
		0x3		

47.4.76 Miscellaneous control signals for in CPU1

This register drives certain input ports of CPU1, providing capability to lock the settings or enhanced security.

Table 963. Miscellaneous control signals for in (CPU1_LOCK_REG, offset = 0xFF0)

Bit	Symbol	Value	Description	Reset value
1:0	LOCK_NS_VTOR		CPU1 VTOR_NS register write-lock	0x2
		0x2	LOCKNSVTOR is 0. Enable writes to the VTOR_NS register	
		0x1	CPU1 LOCKNSVTOR is 1. Disable writes to the VTOR_NS register	
		0x0		
		0x3		

Table 963. Miscellaneous control signals for in (CPU1_LOCK_REG, offset = 0xFF0) *…continued*

Bit	Symbol	Value	Description	Reset value
3:2	LOCK_NS_MPU		CPU1 non-secure MPU register write-lock	0x2
		0x2	CPU1 LOCKNSMPU is 0. Enable writes to non-secure MPU registers	
		0x1	CPU1 LOCKNSMPU is 1. Disable writes to non-secure MPU registers	
		0x0		
		0x3		
29:4	-		Reserved.	undefined
31:30	CPU1_LOCK_REG_LOCK		Disables write access to this register itself. Once locked, only system reset can unlock (to enable write access)	0x2
		0x2	This register can be written.	
		0x1	This register can't be written (included this bitfield).	
		0x0		
		0x3		

47.4.77 Secure control duplicate register

This register is duplicate of **MISC_CTRL_REG**. A secondary register with duplicate programing is implemented to provide better protection against malicious hacking attacks such as glitch attack.

Table 964. Secure control duplicate register (MISC_CTRL_DP_REG, offset = 0xFF8)

Bit	Symbol	Description	Reset value
1:0	WRITE_LOCK	Write lock. 0x2: secure_ctrl_group_rule registers and this register itself can be written. 0x1: secure_ctrl_group_rule registers and this register itself can't be written. Others: Reserved If this field doesn't match **MISC_CTRL_REG** [1:0], both **MISC_CTRL_DP_REG** and **MISC_CTRL_REG** and all secure_ctrl_group_rule registers are write-locked. When the control fields below doesn't match the corresponding fields in the **MISC_CTRL_REG** register, the related control signals are set to the restrictive mode	0x2
3:2	ENABLE_SECURE_CHECKING	Same description as in **MISC_CTRL_REG.**	0x2
5:4	ENABLE_S_PRIV_CHECK	Same description as in	0x2
7:6	ENABLE_NS_PRIV_CHECK	Same description as in	0x2
9:8	DISABLE_VIOLATION_ABORT	Same description as in	0x2
11:10	DISABLE_SIMPLE_MASTER_STRICT_MODE	Same description as in	0x2
13:12	DISABLE_SMART_MASTER_STRICT_MODE	Same description as in	0x2
15:14	IDAU_ALL_NS	Same description as in	0x2
31:16	-	Reserved.	undefined

47.4.78 Secure control register

This register provides more control over certain system behavior as described in individual fields.

Table 965. Secure control register (MISC_CTRL_REG, offset = 0xFFC)

Bit	Symbol	Description	Reset value
1:0	WRITE_LOCK	Write lock. 0x2: secure_ctrl_group_rule registers and this register itself can be written. 0x1: secure_ctrl_group_rule registers and this register itself can't be written. Others: Reserved If this field doesn't match MISC_DP_CTRL_REG [1:0], both MISC_CTRL_DP_REG and MISC_CTRL_REG and all secure_ctrl_group_rule registers are write-locked. When the control fields below doesn't match the corresponding fields in the duplicate register, the related control signals are set to the restrictive mode.	0x2
3:2	ENABLE_SECURE_CHECKING	AHB bus matrix enable secure checking. 0x2: disabled. 0x1: enabled (restrictive mode) Others: Reserved	0x2
5:4	ENABLE_S_PRIV_CHECK	AHB bus matrix enable secure privilege check. 0x2: disabled. 0x1: enabled (restrictive mode) Others: Reserved	0x2
7:6	ENABLE_NS_PRIV_CHECK	AHB bus matrix enable non-secure privilege check. 0x2: disabled. 0x1: enabled (restrictive mode) Others: Reserved	0x2
9:8	DISABLE_VIOLATION_ABORT	Disable secure violation abort. 10: the violation detected by the secure checker causes abort. All other values: the violation detected by the secure checker won't cause abort but secure_violation_irq will still be asserted and serviced by ISR. 01b: disable abort fort secure checker. 10b: enable abort fort secure checker.	0x2
11:10	DISABLE_SIMPLE_MASTER_STRICT_MODE	0x2: Simple master in strict mode. Can read and write to memories at same level only. (Mode recommended by ARM). (restrictive mode) 0x1: Simple master in tier mode. Can read and write to memories at same or below level. It applies to POWERQUAD, DMA0, DMA1, SDIO, USB-FS. Others: Reserved	0x2

Table 965. Secure control register (MISC_CTRL_REG, offset = 0xFFC) *…continued*

Bit	Symbol	Description	Reset value
13:12	DISABLE_SMART_MASTER_STRICT_MODE	0x2: Smart masters in strict mode. Can execute, read and write to memories at same level only. (Mode recommended by ARM.). (restrictive mode) 0x1: Smart masters in tier mode. Can execute at same level only,but read and write to memories at same or below level. signaling is pass through to bus matrix. Others: Reserved	0x2
15:14	IDAU_ALL_NS	0x2: IDAU is enabled. (restrictive mode) 0x1: IDAU is disabled, hence all memories are attributed as non-secure memory.	0x2
31:16	-	Reserved.	undefined

47.4.79 Security configuration

RAM2 region is divided into 16 sub-regions, each rule corresponds to a following sub-region address range: provides ROM support via API to program security registers based on setting in Flash image header. ROM locks the settings before passing control to application code. But more details can be found in secure TZ Boot section.

47.4.80 Hypervisor interrupt

ARMv8-M supervisor call is banked and can therefore exist in secure mode and a separate supervisor handler can exist for Non-secure. Using SVC (Supervisor Call opcode) does not allow Non-secure code to call the Hypervisor because the security attributes of the Hypervisor are secure-privileged and therefore, Non-secure code cannot enter it.

The LPC55S6x offers a hardware implementation whereby Non-secure access of the secure AHB Controller (always tier-4) will raise an interrupt. This interrupt can be configured to be secure. This way Non-secure code can raise secure interrupt and get entry into secure privileged domain. It is used as the call to the Hypervisor.

47.4.81 Authenticated debug access

The SWD Debug supports both secure and Non-secure debug. The LPC55S6x ROM provides support to safeguard secure application code from unauthorized debug access using Debug Authentication process.

The PFR contains fields that allow disabling of secure debug and disabling on NS debug. The anticipated development process is that a secure developer can debug their secure code, then pass the device to an NS developer who can access all NS resources. The NS developer can then disable NS debug, once satisfied all NS code is working, and thus disable all debug via SWD port. After secure debug is disabled and only NS debug is allowed, there is no option to get access to secure resources via the debug port.

The authentication function can be used on both secure and NS debug accesses. See Chapter 51 “LPC55S6x/LPC55S2x/LPC552x Debug Subsystem” for more details on Debug authentication.

47.4.82 TrustZone programming of flash

The LPC55S6x offers two stages of flash programing. First stage is development and deployment. At this stage, JTAG or SWD ports are used for flash erase and programing. However, once device is deployed this mode of flash programing is disabled. Thereafter, only alternative to program flash is over the air via secure BootROM.

47.4.83 Compatibility with ARMv7-M (Cortex-M3/M4)

TrustZone is not the only improvement in ARMv8-M architecture of Cortex-M33. Cortex-M33 has numerous enhancements over ARMv7-M architecture, such as new instructions, DSP engine, upgraded FPU, upgraded MPU and better debug capability. These enhancements enable better software design, making LPC55S6x a great candidate for upgrade. Being 32-bit and thumb-code compatible with existing ARMv7-M architecture, software migration is relatively simple when going from Cortex-M3/M4 to Cortex-M33. The LPC55S6x supports upward transition from Cortex-M4 to Cortex-M33.

Out of reset the M33 CPU will always default to be executing in the S state. However, the BootROM code that executes prior to executing customer reset code, examines the PFR field part config, that will indicate if the part is a TZ-disabled or TZ-enabled.

- If part is TZ-disabled, ROMCode configure the settings to assures that application code developed for CM4 can be used without modifications.
- If part is TZ-enabled, CM33 CPU0 will be in the secure state and will execute secure-privileged application code. Relinquishing of peripherals and their interrupt routing, other bus masters, and regions of RAM to NS is performed at this point, prior to the M33 changing from secure state to NS state.

On the LPC55S6x, CPU1 is Cortex-M33 configuration without TrustZone. This instance always defaults to NS state out of reset and would have no trusted execution support. This mode is backward compatible with ARMv7M CortexM3/M4.

More details can be found in BootROM part configuration chapter. Chapter 6 “LPC55S6x/LPC55S2x/LPC552x Boot ROM”.

48.1 How to read this chapter

The Secure Hash Algorithm (SHA), the Random Number Generator (RNG), AES encryption/decryption registers, the PRINCE real-time encryption/decryption, PUF, DICE, UUID, and security APIs are available on all LPC55S6x/LPC55S2x devices.

48.2 Introduction

The security system on LPC55S6x/LPC55S2x has a set of hardware blocks and ROM code to implement the security features of the device. The hardware consists of an AES engine, a SHA engine, a random number generator, a PRINCE engine, and a key storage block that keys from an SRAM based PUF (Physically Unclonable Function). Figure 183 "Security system" shows an overview of the LPC55S6x/LPC55S2x/LPC552x security system. All components of the system can be accessed by the processor or the DMA engine to encrypt or decrypt data and for hashing. The ROM is responsible for secure boot in addition to providing support for various security functions.

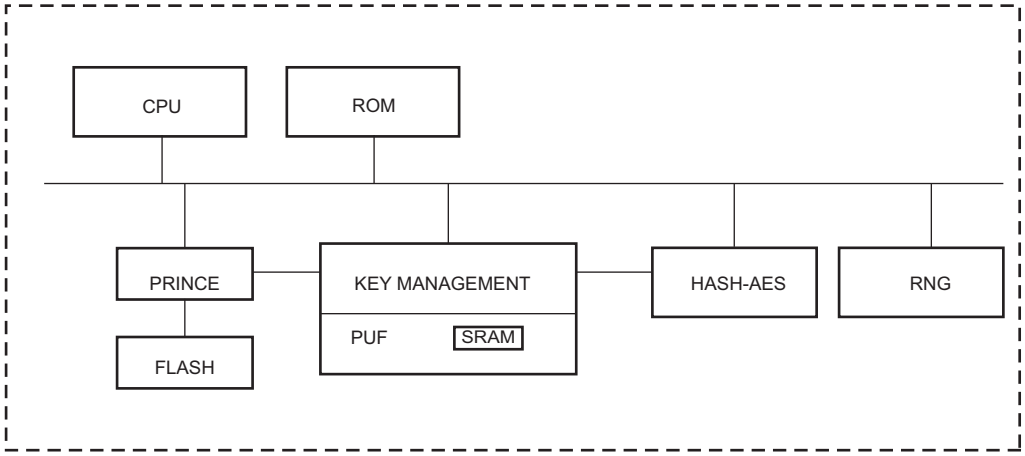


Fig 183. Security system

48.2.1 Key storage/management

A critical feature of any security system is how keys are stored and managed. Keys can be used for boot loading and handling of critical user data. LPC55S6x/LPC55S2x offers SRAM PUF where the PUF provides a unique key per device and exists in that device based on the unique characteristics of PUF SRAM. Figure 184 "Key storage" shows the block diagram of how keys are used by the AES engine. PUF keys have a dedicated path to the AES engine and PRINCE engine where only intended engine can make use of its key. There is no other mechanism by which keys can be observed. KEY0 feeds into AES, KEY1/2/3 from Key Management block feed into PRINCE.

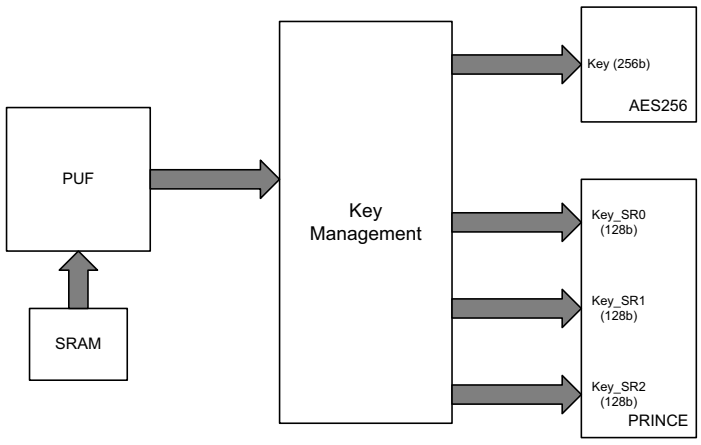


Fig 184. Key storage

48.2.1.1 PUF keys

The PUF controller provides secure key storage without storing the key. It is done by using the digital fingerprint of a device derived from SRAM. Instead of storing the key, a key code is generated, which in combination with the digital fingerprint is used to reconstruct keys that are routed to the AES engine or for use by software. The PUF controller provides generation and secure storage for keys.

48.3 AES engine

The LPC55S6x/LPC55S2x devices provide an on-chip hardware AES encryption and decryption engine to protect the image content and to accelerate processing for data encryption or decryption, data integrity, and proof of origin. Data can be encrypted or decrypted by the AES engine using a key from the PUF or a software supplied key.

See Section 48.12 “AES engine functional details”

ICB-AES mode exists to provide a counter measure to Side Channel Analysis. It is slower in exchange for this counter measure protection

48.4 SHA

All LPC55S6x/LPC55S2x devices provide on-chip Hash support to perform SHA-1 and SHA-2 with 256-bit digest (SHA-256). Hashing is a way to reduce arbitrarily large messages or code images to a relatively small fixed size “unique” number called a digest. The SHA-1 Hash produces a 160 bit digest (five words), and the SHA-256 hash produces a 256 bit digest (eight words).

For the SHA hardware:

- Even a small change to the input message will cause a major change in the digest output. Therefore, for a given input message or image there is only one digest.
- There is no predictable way to modify one input to result in a specific digest. A message cannot be added, inserted, or modified to get the same Hash in any direct way.

These two properties make it useful for verifying a message is valid, or corrupted intentionally or unintentionally.

Hashing is used for four primary purposes:

- Core of a digital signature model, including certificates, for example and for secure update.
- Support a challenge/response or to validate a message when used with a Hash-based Message Authentication Code (HMAC).
- In a secure boot model, which verifies code integrity.
- Verify external memory that has not been compromised.

See Section 48.13 “HASH functional details”.

48.5 Digital signatures

A digital signature combines public/private keys such as RSA or ECC with SHA Hashing. Signature is formed in the following way:

- The message or image is hashed using SHA1, SHA2, or other.
- The digest is formed into a buffer along with a header and padding. The header indicates what hashing was used (for example,SHA1). The padding fits the buffer to the size of the public key, for example, 256-bits, 1024-bits, and 2048-bits.
- The buffer is signed (encrypted) using the private key. Note: That signing is a reverse of the normal public or private key where anyone can encrypt using the public key and therefore, only the private key holder can decrypt. In this case, the model is reversed so that only a private key holder can sign, and anyone can verify it if it is from the private key holder.

To verify the signature, the following steps are used:

- The message or image is hashed using SHA1, SHA2. The output of the hash must match the signature.
- The signature is decrypted using the public key.
- The hash digest is compared against the stored digest in the buffer after decryption.

The advantage of the signature model is that the public key and the signature can be public. Therefore, the signature can be stored in an external flash file along with the image and then verified using a stored copy of the public key.

48.6 Hash-based Message Authentication Code (HMAC)

An HMAC can be achieved on LPC55S6x/LPC55S2x using a pre-shared key and hashing. It is a way of verifying a message (encrypted or not) and can also be used for challenge or response. Both sides must have a pre-shared key that is just a shared secret value and not an encryption key.

The HMAC is formed using three steps:

- Hashing the message or image.
- Taking the resultant digest and combine with the key and some padding.
- Hashing the combination of digest, key, and padding. The resultant digest is sent.

On the other side, the same procedure is followed to verify and get the same digest. Only those parties that know the key can get the correct digest and therefore, can trust the data that was hashed.

HMACs are significantly faster than signatures, but work only with pre-shared keys, which must not be leaked or lost (unlike a public key). The HMAC key can be shared dynamically using trust models like Diffie-Hellman or maybe a board-unique key shared by two devices.

48.7 RNG

Random Number Generators (RNG) are used for cryptographic, modeling, and simulation applications, which employ keys that must be generated in a random fashion.

See Section 48.15 “RNG functional details”.

48.8 UUID

LPC55S6x/LPC55S2x/LPC552x store 128-bit IETF RFC4122 compliant non-sequential Universally Unique Identifier (UUID). It can be read from flash PFR region at register location 0x0009_FC70 onwards.

48.9 DICE

LPC55S6x/LPC55S2x (secure part) supports Device Identifier Composition Engine (DICE) to provide Composite Device Identifier (CDI). CDI value would be available at SYSCON offset 0x0900 to 0x091C for consumption after boot completion. It is recommended to overwrite these registers once ephemeral key-pairs are generated using this value.

Note: If using DICE, it is recommended to use a secure provisioning method, otherwise it is possible to allow for information leakage of parts of the CDI.

48.10 PRINCE real-time encryption/decryption

LPC55S6x/LPC55S2x devices offer support for real-time encryption and decryption for on-chip flash using the PRINCE encryption algorithm. Compared to AES, PRINCE is fast because it can decrypt and encrypt without adding extra latency. PRINCE operates as data is read or written, without the need to first store data in RAM and then encrypt or decrypt to another space. It operates on a block size of 64-bits with a 128-bit key.

This functionality is useful for asset protection, such as securing application code, securing stored keys, and enabling secure flash update.

See Section 48.17.1 “Functional details”

48.11 PUF controller and key management

The PUF controller provides a secure key storage without injecting or provisioning device unique PUF root key. See Section 48.2.1.1 “PUF keys” for more details.

48.11.1 PUF controller features

The PUF controller has the following features:

- Key strength of 256-bits.
  - The PUF constructs 256-bit strength device unique PUF root key using the digital fingerprint of a device derived from SRAM and error correction data called Activation Code (AC). The AC is generated during enrollment process and must be stored on external non-volatile memory device in the system.
- Generation, storage, and reconstruction of keys.
- Key sizes from 64-bits to 4096-bits.
  - PUF controller allows storage of keys, generated externally or on chip, of sizes 64-bits to 4096-bits.
  - PUF controller combines keys with digital fingerprint of device to generate key codes. These key codes should be provided to the controller to reconstruct original key. They can be stored on external non-volatile memory device in the system.
- Key output via dedicated hardware interface or through register interface.
  - PUF controller allows to assign a 4-bit index value for each key while generating key codes. Keys that are assigned index value zero are output through HW bus, accessible to AES and PRINCE engines only. Keys with non-zero index are available through APB register interface.
- 32-bit APB interface.
- Programmable feature to block indices from generating new key codes.

48.11.2 Basic configuration

- The PUF block can be reset using the PUF_RST bit in PRESETCTRL2 register. See Table 47. However, certain registers in key management wrapper are reset only on global system reset, not on this IP reset.

- The clock to the PUF can be controlled using the PUF bit in AHBCLKCTRL2 register. See Table 57.

48.11.3 PUF controller operations

The PUF controller supports the following operations:

1. Enroll: The controller retrieves the Startup Data (SD) from the memory (SRAM), derives a digital fingerprint, generates the corresponding Activation Code (AC) and sends it to the Client Design (CD) to be stored externally. Perform this step only once for each device.

   There is a configuration register CFG that can block further enrollment. This register is R/W1S and is cleared by reset.
2. Start: The AC generated during the enroll operation and the SD are used to reconstruct the digital fingerprint. It is done after every power-up and reset.
3. Generate Key: The controller generates an unique key and combines it with the digital fingerprint to output a key code.

   Each time a Generate Key operation is executed a new unique key is generated.
4. Set Key: The digital fingerprint generated during the Enroll/Start operations and the key provided by the Client Design (CD) are used to generate a Key Code. This KC can be stored externally. Perform this operation only once for each key.
5. Get Key: The digital fingerprint generated during the Start operation and the KC generated during a Set Key operation are used to retrieve a stored key. Perform this operation every time a key is needed.

48.11.4 SRAM PUF power control

The SRAM used for the PUF is a separate block of memory that is only accessible by PUF controller. All self-test is built into the PUF logic and is not accessible through any chip test modes. The PUF SRAM has a power switch to ensure there is clean start-up power and is controlled via the PUF PWRCTRL register (address, 0x4003 B108). If a previously powered PUF SRAM is turned off by clearing the bit RAMON in PWRCTRL register, up to 400 ms second delay is required before the PUF SRAM can be powered again by setting bit RAMON to 1. The required delay depends on temperature conditions and is highest only at the worst corners. ROM API implements a special scheme where the first attempt is made with smaller delay and if that fails, a 400 ms delay is applied in the next iteration.

48.11.5 Key management

The LPC55S6x/LPC55S2x key management module supports storing an AES Key (KEY0) and three PRINCE Keys (KEY1, KEY2, KEY3). These keys are fed into their respective IPs via a dedicated hard-wired interface and are not readable by software. Since these keys are from Index-0, they are inaccessible by the software interface. The PRINCE requires a 128-bit key. The LPC55S6x/LPC55S2x device supports up to three regions, therefore, three separate 128-bit keys are made available via the key management block. The AES key can be 128-bits, 192-bits or 256-bits in length.

PUF keys for AES and PRINCE, if already loaded, are retained during deep-sleep and power-down but not retained during deep-power down. The CTRL, CFG, KEYLOCK, KEYENABLE, KEYRESET, IDBLK_L/H, IDXBLK__DP/H_DP, SHIFT_STATUS registers are preserved and not reset during power-down.

This module supports blocking of access to a set of indexes such that they cannot be used anymore for key generation or retrieval until next reset.

48.11.5.1 Key loading procedure

To load KEYn for use by the AES or PRINCE, use the following procedure:

1. Write the enable value, 0x2, to the KEYn field of the KEYRESET register, to clear the associated KEYn hold and KEYn_SHIFT_STATUS registers.
2. Write the enable value, 0x2, to the KEYn field of the KEYENABLE register. Ensure that only the intended KEYn field in KEYENABLE register is enabled. The other KEYn fields should be disabled to avoid overwriting other keys.
3. For added security protection, write a random mask value to the KEYMASKn register.
4. Issue the Get Key command to the PUF, requesting the desired key with KEYINDEX=0, so that the key is presented on the dedicated hardware interface to the key management module. See sections Section 48.11.7.11 “Get Key” and Section 48.11.8.6 “Pseudocode Get Key function” function. It is assumed that PUF initialization and start have already been performed before issuing Get Key. The requested key will be loaded into the KEYn hold register, which is only visible to the AES or PRINCE.
5. If required, write the disable value, 0x1, to the KEYn field of the KEYLOCK register, to prevent any further changes to KEYn.

48.11.6 Register interface

Table 966 shows the registers and their addresses.

Table 966. PUF controller registers (base address = 0x4003 B000)

Name	Access	Address	Description	Reset value	Section
CTRL	R/W	0x00	PUF control register.	0x0	Section 48.11.6.1
KEYINDEX	R/W	0x04	PUF key index register.	0x0	Section 48.11.6.2
KEYSIZE	R/W	0x08	PUF key size register.	0x0	Section 48.11.6.3
STAT	R	0x20	PUF status register.	0x00000001	Section 48.11.6.4
ALLOW	R	0x28	PUF allow register.	0x0	Section 48.11.6.5
KEYINPUT	W	0x40	PUF key input register.	0x0	Section 48.11.6.6
CODEINPUT	W	0x44	PUF code input register.	0x0	Section 48.11.6.7
CODEOUTPUT	R	0x48	PUF code output register.	0x0	Section 48.11.6.8
KEYOUTINDEX	R	0x60	PUF key output index register.	0x0	Section 48.11.6.9
KEYOUTPUT	R	0x64	PUF key output register.	0x0	Section 48.11.6.10
IFSTAT	R/W1C	0xDC	PUF interface status and clear.	0x0	Section 48.11.6.11
VERSION	R	0xFC	PUF version register.	0x0	Section 48.11.6.12
INTEN	R/W	0x100	Interrupt enable.	0x0	Section 48.11.6.13
INTSTAT	RO/W1C	0x104	Interrupt status	0x0	Section 48.11.6.14
PWRCTRL	R/W	0x108	PUF RAM power control.	0x0	Section 48.11.6.15
CFG	R/W1S	0x10C	Configuration register for block bits.	0x0	Section 48.11.6.16
-	-	0x110-0x1FC	Reserved.	0x0	
KEYLOCK	RW	0x200	Key lock register.	0x000000AA	Section 48.11.6.17

Table 966. PUF controller registers *…continued* **(base address = 0x4003 B000)** *…continued*

Name	Access	Address	Description	Reset value	Section
KEYENABLE	RW	0x204	Key enable register.	0x00000055	Section 48.11.6.18
KEYRESET	WO	0x208	Key reset register.	0x0	Section 48.11.6.19
IDXBLK_L	RW	0x20C	IDXBLK_L register.	0x8000AAAA	Section 48.11.6.20
IDXBLK_H_DP	RW	0x210	IDXBLK_H_DP register.	0x8000AAAA	Section 48.11.6.21
KEYMASK0	WO	0x214	Key mask0 register.	0x0	Section 48.11.6.22
KEYMASK1	WO	0x218	Key mask1 register.	0x0	Section 48.11.6.22
KEYMASK2	WO	0x21C	Key mask2 register.	0x0	Section 48.11.6.22
KEYMASK3	WO	0x220	Key mask3 register.	0x0	Section 48.11.6.22
-		0x224-0x250	Reserved.	0x0	
IDXBLK_H	RW	0x254	IDXBLK_H register.	0x8000AAAA	Section 48.11.6.23
IDXBLK_L_DP	RW	0x258	IDXBLK_L_DP register.	0x8000AAAA	Section 48.11.6.24
SHIFT_STATUS	R	0x25C	Shift status register.	0x0	Section 48.11.6.25

In the following sections, the PUF register bits are defined.

48.11.6.1 PUF control register

The PUF control register defines which command must be executed next. The bits automatically revert to 0. Only one command bit may be written with 1 at a time, with the exception of ZEROIZE. Writing ZEROIZE with 1 takes precedence over all other commands.

Table 967. PUF control register (CTRL, offset = 0x00)

Bit	Symbol	Description	Reset value
0	ZEROIZE	Begin Zeroize operation for PUF and go to error state.	0
1	ENROLL	Begin Enroll operation.	0
2	START	Begin Start operation.	0
3	GENERATEKEY	Begin Generate Key operation.	0
4	SETKEY	Begin Set Key operation.	0
6	GETKEY	Begin Get Key operation.	0
31:7	-	Reserved. Read value is 0, only 0 should be written.	0

48.11.6.2 PUF key index register

The PUF key index register defines the key index for the next set key operation.

Table 968. PUF key index register (KEYINDEX, offset = 0x04)

Bit	Symbol	Description	Reset value
3:0	KEYIDX	Key index for Set Key operations.	0
31:4	-	Reserved. Read value is 0, only 0 should be written.	0

48.11.6.3 PUF key size register

The PUF key size register defines the key index for the next set key operation.

Table 969. PUF key size register (KEYSIZE, offset = 0x08)

Bit	Symbol	Description	Reset value
5:0	KEYSIZE	Key size for Set Key operations.	0
31:6	-	Reserved. Read value is 0, only 0 should be written.	0

Coding of the KEYSIZE filed is defined in Section 48.11.7.3 “Key and code sizes”.

48.11.6.4 PUF status register

The PUF status register indicates the current status of the PUF and indicates which data is requested or available.

Table 970. PUF status register (STAT, offset = 0x20)

Bit	Symbol	Description	Reset value
0	BUSY	Indicates that operation is in progress.	1
1	SUCCESS	Last operation was successful.	0
2	ERROR	PUF is in the error state and no operations can be performed.	0
3	-	Reserved. Read value is undefined, only 0 should be written.	0
4	KEYINREQ	Request for next part of key.	0
5	KEYOUTAVAIL	Next part of key is available.	0
6	CODEINREQ	Request for next part of AC/KC.	0
7	CODEOUTAVAIL	Next part of AC/KC is available.	0
31:8	-	Reserved. Read value is 0, only 0 should be written.	0

The indicated reset value is present immediately after reset. After the PUF finishes initialization, the BUSY bit goes to 0 and depending on the state of the PUF, either the SUCCESS bit or the ERROR bit goes to 1. See Section 48.11.7.1 “Order of operations”.

48.11.6.5 PUF allow register

The PUF allow register indicates which operations are currently allowed.

Table 971. PUF allow register (ALLOW, offset = 0x28)

Bit	Symbol	Description	Reset value
0	ALLOWENROLL	Enroll operation is allowed.	0
1	ALLOWSTART	Start operation is allowed.	0
2	ALLOWSETKEY	Set Key operations are allowed.	0
3	ALLOWGETKEY	Get Key operation is allowed.	0
31:4	-	Reserved. Read value is 0, only 0 should be written.	0

The indicated reset value is present immediately after reset. After the PUF finishes initialization, one or more bits of this register goes to 1.

48.11.6.6 PUF key input register

The PUF reads the key that must be stored during the Set Key operation using the PUF key input register.

Table 972. PUF key input register (KEYINPUT, offset = 0x40)

Bit	Symbol	Description	Reset value
31:0	KEYIN	Key input data. This field must only be written when KEYINREQ = 1.	0

48.11.6.7 PUF code input register

The PUF reads the AC (in case of a start operation) or the KC (in case of a Get Key operation) using the PUF code input register.

Table 973. PUF code input register (CODEINPUT, offset = 0x44)

Bit	Symbol	Description	Reset value
31:0	CODEIN	AC/KC input data. This field must only be written when CODEINREQ = 1.	0

48.11.6.8 PUF code output register

The PUF provides the AC (in case of an enroll operation) or KC (in case of a Set Key or Generate Key operation) using the PUF code output register.

Table 974. PUF code output register (CODEOUTPUT, offset = 0x48)

Bit	Symbol	Description	Reset value
31:0	CODEOUT	AC/KC output data. This field must only be written when CODEOUTAVAIL = 1.	0

48.11.6.9 PUF key output index register

The key index of the reconstructed key can be read using the PUF key output index register.

Table 975. PUF output index register (KEYOUTINDEX, offset = 0x60)

Bit	Symbol	Description	Reset value
3:0	KEYOUTIDX	Key index for the key that is currently output using the key output register.	0
31:4	-	Reserved. Read value is 0, only 0 should be written.	0

48.11.6.10 PUF key output register

The reconstructed key can be read using the PUF key output register.

Table 976. PUF output index register (KEYOUTPUT, offset = 0x64)

Bit	Symbol	Description	Reset value
31:0	KEYOUT	Key output data. This field must only be read when KEYOUTAVAIL= 1	0

48.11.6.11 PUF interface status register

The status of the APB interface can be monitored with the PUF interface status register. This register has the same address as IFSTATCLR.

Table 977. PUF interface status register (IFSTAT: offset = 0xDC)

Bit	Symbol	Description	Reset value
0	ERROR	Read: indicates that any of the following errors have occurred: Write to a non-existing register. Read from a non-existing register. Write to a read-only register. Read from a write-only register. KEYINPUT register is written when no key is requested (KEYINREQ). CODEINPUT register is written when no AC/KC is requested (CODEINREQ = 0). CODEOUTPUT register is read when no AC/KC is available (CODEOUTAVAI = 0). KEYOUTPUT register is read when no key is available (KEYOUTAVAIL = 0). KEYOUTINDEX register is read when no key is available (KEYOUTAVAIL = 0). Multiple commands are written at the same time to the PUF control register. A command is written that is not allowed. Write: writing a 1 clears the error flag.	0
31:1	-	Reserved. Read value is 0, only 0 should be written.	-

48.11.6.12 PUF version register

The PUF version register provides the version of the PUF module.

Table 978. PUF version register (VERSION, offset = 0xFC)

Bit	Symbol	Description	Reset value
31:0	VERSION	Version of the PUF module.	0

48.11.6.13 PUF interrupt enable register

The PUF interrupt enable register is used to enable various PUF controller interrupt sources. Enable bits in INTEN are mapped in locations that correspond to the flags in the STAT register.

Table 979. PUF interrupt enable register (INTEN, offset = 0x100)

Bit	Symbol	Description	Reset value
0	READYEN	Indicates that the initialization or a operation is completed.	0
1	SUCCESEN	Last operation was successful.	0
2	ERROREN	PUF is in the error state and no operations can be performed.	0
3	-	Reserved. Read value is 0, only 0 should be written.	0
4	KEYINREQEN	Request for next part of key.	0
5	KEYOUTAVAILEN	Next part of key is available.	0
6	CODEINREQEN	Request for next part of AC/KC.	0
7	CODEOUTAVAILEN	Next part of AC/KC is available.	0
31:8	-	Reserved. Read value is 0, only 0 should be written.	0

48.11.6.14 PUF interrupt status register

The PUF interrupt status register provides a view of interrupt flags that are currently enabled.

Table 980. PUF interrupt status register (INTSTAT, offset = 0x104)

Bit	Symbol	Description	Reset value
0	READY	Indicates that the initialization or a operation is completed. Write 1 to clear.	0
1	SUCCESS	Last operation was successful. Cleared when interrupt source clears.	0
2	ERROR	PUF is in the error state (for example, an incorrect key code, or Zeroization) and no operations can be performed.	0
3	-	Reserved. Read value is 0, only 0 should be written.	-
4	KEYINREQ	Request for next part of key. Cleared when interrupt source clears.	0
5	KEYOUTAVAIL	Next part of key is available. Cleared when interrupt source clears.	0
6	CODEINREQ	Request for next part of AC/KC. Cleared when interrupt source clears.	0
7	CODEOUTAVAIL	Next part of AC/KC is available. Cleared when interrupt source clears.	0
31:8	-	Reserved. Read value is 0, only 0 should be written.	0

48.11.6.15 PUF power control register

The PUF power register controls the power of the dedicated SRAM used by PUF controller.

Table 981. PUF power control register (PWRCTRL, offset = 0x108)

Bit	Symbol	Description	Reset value
0	RAMON	Power on the PUF RAM. This bit is cleared by hardware when a low power mode is entered.	0
1	RAMSTAT	PUF RAM status. This bit is read only. It is set to 1 when PUF RAM completes initialization after RAMON bit is set and cleared when RAMON bit is cleared.	
31:2	-	Reserved. Read value is undefined, only 0 should be written.	0

48.11.6.16 PUF configuration register

The PUF configuration register allows blocking of enrollment and additional key code generations.

Table 982. PUF configuration register (CFG, offset = 0x10C)

Bit	Symbol	Description	Reset value
0	BLOCKENROLL_SETKEY	Block enroll and set key operation. Write 1 to set, cleared on reset.	0
1	BLOCKKEYOUTPUT	Block key output data. If the bit is set to 1, it will block key output (key out data = 0) when key output index = 15. This bit is cleared on reset. If the bit is set to 0, then key out will not be blocked, even if key output index = 15.	0
31:2	-	Reserved. Read value is undefined, only 0 should be written.	0

48.11.6.17 Key lock register

The PUF key lock register allows locking write access to a set of registers associated with a given key in Key management module. Using this feature, user have option of locking the key settings once key loading is completed.

Table 983. Key lock register (KEYLOCK, offset = 0x200)

Bit	Symbol	Value	Description	Reset value
1:0	KEY0			0x2
		0x2	Write access to KEY0MASK, KEYENABLE.KEY0 and KEYRESET.KEY0 is allowed.	
		0x1	Write access to KEY0MASK, KEYENABLE.KEY0 and KEYRESET.KEY0 is NOT allowed. Once 0x1 is written in this field, its value cannot be modified until a POR occurs.	
		Others	Reserved.	
3:2	KEY1			0x2
		0x2	Write access to KEY1MASK, KEYENABLE.KEY1 and KEYRESET.KEY1 is allowed	
		0x1	Write access to KEY1MASK, KEYENABLE.KEY1 and KEYRESET.KEY1 is NOT allowed. Once 0x1 is written in this field, its value cannot be modified until a POR occurs.	
		Others	Reserved.	
5:4	KEY2			0x2
		0x2	Write access to KEY2MASK, KEYENABLE.KEY2 and KEYRESET.KEY2 is allowed.	
		0x1	Write access to KEY2MASK, KEYENABLE.KEY2 and KEYRESET.KEY2 is NOT allowed. Once 0x1 is written in this field, its value cannot be modified until a POR occurs.	
		Others	Reserved.	
7:6	KEY3			0x2
		0x2	Write access to KEY3MASK, KEYENABLE.KEY3 and KEYRESET.KEY3 is allowed.	
		0x1	Write access to KEY3MASK, KEYENABLE.KEY3 and KEYRESET.KEY3 is NOT allowed. Once 0x1 is written in this field, its value cannot be modified until a POR occurs.	
		Others	Reserved.	
31:8	-	-	Reserved.	Undefined

48.11.6.18 Key enable register

The PUF key enable register allows user to load PUF output as secret key to a particular engine.

Table 984. Key enable register (KEYENABLE, offset = 0x204)

Bit	Symbol	Access	Value	Description	Reset value
1:0	KEY0	RW			0x1
			0x2	Data coming out from PUF Index 0 interface are shifted in KEY0 register.	
			0x1	Data coming out from PUF Index 0 interface are NOT shifted in KEY0 register.	
			Others	Reserved.	
3:2	KEY1	RW			0x1
			0x2	Data coming out from PUF Index 0 interface are shifted in KEY1 register.	
			0x1	Data coming out from PUF Index 0 interface are NOT shifted in KEY1 register.	
			Others	Reserved.	
5:4	KEY2	RW			0x1
			0x2	Data coming out from PUF Index 0 interface are shifted in KEY2 register.	
			0x1	Data coming out from PUF Index 0 interface are NOT shifted in KEY2 register.	
			Others	Reserved.	
7:6	KEY3	RW			0x1
			0x2	Data coming out from PUF Index 0 interface are shifted in KEY3 register.	
			0x1	Data coming out from PUF Index 0 interface are NOT shifted in KEY3 register.	
			Others	Reserved.	
31:8	-	RW		Reserved.	undefined

48.11.6.19 Key reset register

The PUF key reset register allows user to reset Hold register that holds an individual key as well as associated field in SHIFT_STATUS register.

Table 985. Re-initialize keys shift registers counters (KEYRESET, offset = 0x208)

Bit	Symbol	Access	Value	Description	Reset value
1:0	KEY0	WO	0x2	Reset KEY0 hold register and KEY0_SHIFT_STATUS. Self clearing. Must be done before loading any new key.	0x0
			Others	Reserved.	
3:2	KEY1	WO	0x2	Reset KEY1 hold register and KEY1_SHIFT_STATUS. Self clearing. Must be done before loading any new key.	0x0
			Others	Reserved.	
5:4	KEY2	WO	0x2	Reset KEY2 hold register and KEY2_SHIFT_STATUS. Self clearing. Must be done before loading any new key.	0x0
			Others	Reserved.	

Table 985. Re-initialize keys shift registers counters (KEYRESET, offset = 0x208) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
7:6	KEY3	WO	0x2	Reset KEY3 hold register and KEY3_SHIFT_STATUS. Self clearing. Must be done before loading any new key.	0x0
			Others	Reserved.	
31:8	-	WO		Reserved.	0x0

48.11.6.20 Index blocking register (IDX1 - IDX7)

The PUF index blocking register allows user to block a given index from PUF index 1-7. With IDXn bit set, Key output from that index is not available on APB register interface. Index blocking would be activated if relevant key fields in IDXBLK_L and IDXBLK_L_DP do not match. For example, IDX2 would only be accessible if IDX2 = 0x2 in both IDXBLK_L and IDXBLK_L_DP registers.

Table 986. (IDXBLK_L, offset = 0x20C)

Bit	Symbol	Access	Value	Description	Reset value
1:0	-	-	-	Reserved	0
3:2	IDX1	RW		Blocks PUF index 1.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
5:4	IDX2	RW		Blocks PUF index 2.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
7:6	IDX3	RW		Blocks PUF index 3.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
9:8	IDX4	RW		Blocks PUF index 4.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
11:10	IDX5	RW		Blocks PUF index 5.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
13:12	IDX6	RW		Blocks PUF index 6.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	

Table 986. (IDXBLK_L, offset = 0x20C) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
15:14	IDX7	RW		Blocks PUF index 7.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
29:16	-	-		Reserved.	undefined
31:30	LOCK_IDX	WO		Locks 0 to 7 PUF key indexes.	0x2
			0x2	Write access to this register is enabled.	
			0x1	Write access to this register is locked. Once 0x1 is written in this field, its value cannot be modified until a Power On Reset (PoR) occurs.	
			Others	Reserved.	

48.11.6.21 Index blocking duplicate register (IDX8 - IDX15)

This register is duplicate of IDXBLK_H register and provides protection against malicious attacks. Index blocking is activated if relevant key fields in IDXBLK_H and IDXBLK_H_DP do not match. For example, IDX12 is accessible if IDX12 = 0x2 in both IDXBLK_H and IDXBLK_H_DP registers.

Table 987. (IDXBLK_H_DP, offset = 0x210)

Bit	Symbol	Access	Value	Description	Reset value
1:0	IDX8	RW		Blocks PUF index 8.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
3:2	IDX9	RW		Blocks PUF index 9.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
5:4	IDX10	RW		Blocks PUF index 10.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
7:6	IDX11	RW		Blocks PUF index 11.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
9:8	IDX12	RW		Blocks PUF index 12.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	

Table 987. (IDXBLK_H_DP, offset = 0x210) *...continued*

Bit	Symbol	Access	Value	Description	Reset value
11:10	IDX13	RW		Blocks PUF index 13.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
13:12	IDX14	RW		Blocks PUF index 14.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
15:14	IDX15	RW		Blocks PUF index 15.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
31:16	-	-		Reserved.	0x0

48.11.6.22 Key mask register

This registers is additional protection against Side Channel analysis. It obscures the secret key value stored in key hold registers. A random value can be loaded into this register. This register resets in case of a full IC reset.

Only reset in case of full IC reset (KEYMASK [0:3], offset 0x214 - 0x20)

Bit	Symbol	Access	Value	Description	Reset value
31:0	KEYMASK	WO		Reserved.	0x0

48.11.6.23 IDXBLK_H register

The PUF index blocking register allows the user to block a given index from PUF index 8-15. With IDXn bit set, the Key output from that index is not available on APB register interface.

Table 988. (IDXBLK_H, offset = 0x254)

Bit	Symbol	Access	Value	Description	Reset value
1:0	IDX8	RW		Blocks PUF index 8.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
3:2	IDX9	RW		Blocks PUF index 9.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	

Table 988. (IDXBLK_H, offset = 0x254) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
5:4	IDX10	RW		Used to block PUF index 10.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
7:6	IDX11	RW		Blocks PUF index 11.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
9:8	IDX12	RW		Blocks PUF index 12.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
11:10	IDX13	RW		Blocks PUF index 13.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
13:12	IDX14	RW		Blocks PUF index 14.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
15:14	IDX15	RW		Blocks PUF index 15.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
29:16	-	WO		Reserved.	undefined
31:30	LOCK_IDX	RW		Locks 8 to 15 PUF key indexes. Once 0x1 is written in this field, its value cannot be modified until a POR occurs.	0x2

48.11.6.24 IDXBLK_L_DP register

This register is duplicate of IDXBLK_L register and provides protection against malicious attacks. Index blocking would be activated if relevant key fields in IDXBLK_L and IDXBLK_L_DP do not match. For example, IDX4 is accessible if IDX4 = 0x2 in both IDXBLK_L and IDXBLK_L_DP registers.

Table 989. (IDXBLK_L_DP, offset = 0x258)

Bit	Symbol	Access	Value	Description	Reset value
1:0	IDX0	RW		Blocks PUF index 0.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	

Table 989. (IDXBLK_L_DP, offset = 0x258) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
3:2	IDX1	RW		Blocks PUF index 1.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
5:4	IDX2	RW		Blocks PUF index 2.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
7:6	IDX3	RW		Blocks PUF index 3.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
9:8	IDX4	RW		Blocks PUF index 4.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
11:10	IDX5	RW		Blocks PUF index 5.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
13:12	IDX6	RW		Blocks PUF index 6.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
15:14	IDX7	RW		Blocks PUF index 7.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
31:16		WO		Reserved.	undefined

48.11.6.25 SHIFT_STATUS register

This register describes number of words loaded into the key hold register. User can rely on this register to assure that correct number of words are loaded for a given crypto engine before enabling encryption/decryption.
PRINCE requires 128-bit secret key, hence, four 32-bit words must be loaded before starting PRINCE operation. Similarly for AES128, four words must be loaded, for AES192 six words must be loaded, or for AES256 8 words must be loaded before starting AES operation.

Table 990. (SHIFT_STATUS, offset = 0x25C)

Bit	Symbol	Access	Value	Description	Reset value
3:0	KEY0	RO		Index counter from key 0 hold register.	0x0
7:4	KEY1	RO		Index counter from key 1 hold register.	0x0
11:8	KEY2	RO		Index counter from key 2 hold register.	0x0
15:12	KEY3	RO		Index counter from key 3 hold register.	0x0
31:16		RO		Reserved.	undefined

48.11.7 Using PUF

This section describes steps for setting up and usage of PUF block.

48.11.7.1 Order of operations

After power-up or reset the PUF controller starts in one of the three Init states, depending on its previous state. See Figure 185. It first initializes itself (indicated by BUSY = 1).

When initialization is finished, the PUF controller can be moved to one of the cold or warm states. After power-up an enroll or a start operation can be performed.

Note: The enroll operation can only be performed when BLOCKENROLL = 0. If an error occurs during enrollment, the PUF controller goes to an error state.

After enrollment, only the Set Key operations can be performed. These operations can be performed repeatedly. When the device is reset from the enrolled state the PUF controller goes to the error state and no new operations can be performed. New operations can be performed only after re-powering the device.

After the start operation Set Key and Get Key operations can be performed. The Set Key and Get Key operations can be performed repeatedly. When the device is reset the Start operation must be performed again, before performing Get Key and Set Key operations.

Note: The Generate Key and Set Key operations can only be performed when BLOCKSETKEY = 0.

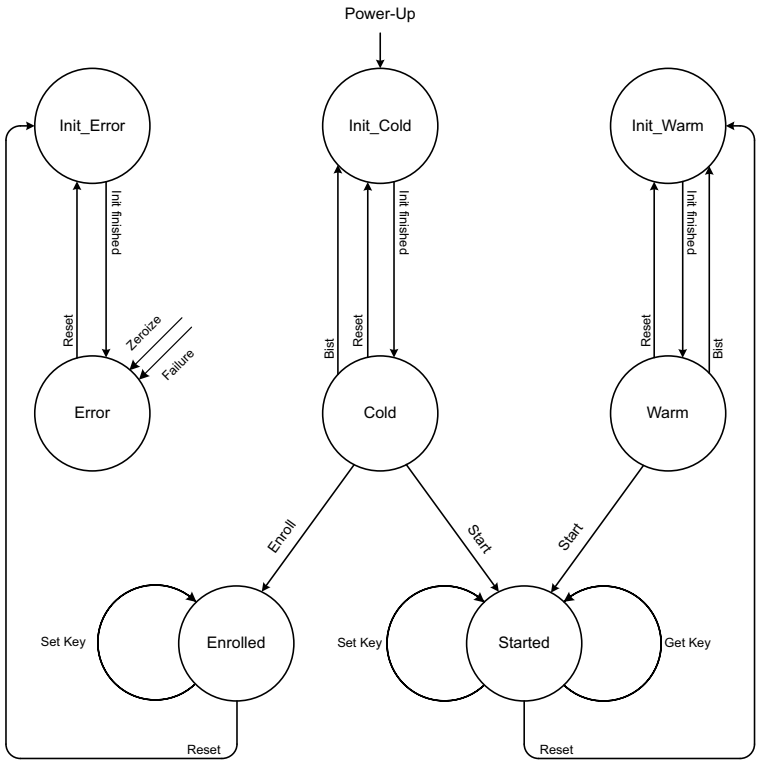


Fig 185. Possible flows, states, and actions

In case of a ZEROIZE operation through the control register or a failure for example, a wrong activation code the PUF controller goes to the error state. It erases all internal critical security parameters and disables communication with the PUF. The only way to leave this state is by repowering the device.

The indicative length of each operation is shown in Table 991, assuming that data is available when requested and data can be accepted when presented by the PUF controller. The number of clock cycles may vary because of internal runtime variations, even when running the same operation with the same data.

Table 991. Number of clock cycles per operation

Operation	Number of clock cycles
Initialization	46.2 k
Enroll	17.1 k
Start	37.0 k
Generate Key (128-bit)	1.8 k
Generate Key (256-bit)	1.8 k
Set Key (128-bit)	2.0 k
Set Key (256-bit)	2.0 k
Get Key (128-bit)	2.1 k
Get Key (256-bit)	2.1 k

48.11.7.2 Activation code size

The size of the Activation Code (AC size) is:

```
AC size = 9536 bits (1192 bytes)
```

48.11.7.3 Key and code sizes

Keys are protected using the digital fingerprint, which has a 256-bit key strength. Longer keys can be stored for cryptographic purposes. For example, ECC keys up to 512-bits or RSA keys up to 4096-bit can be stored safely.

Note: Keys generated by the PUF controller are by construction randomly generated. It means that generated keys cannot be used for cryptographic algorithms that require keys with a specific mathematical structure, which is typical for public key schemes like RSA. In such cases, an externally Generated key should be used and stored as a user key.

The Key Code size (KC size in bits) depends on the key size and can be calculated as:

```
(160u + (((Key size + 255u) / 256) x 256))
```

Table 992 specifies the KEYSIZE values to use for the supported key sizes, and the size of the related PUF-generated KC.

Table 992. Coding of KEYSIZE

KEYSIZE (5:0)	Value	Key size (bits)	KC size (bits)	KC size (bytes)
000001	1	64	416	52
000010	2	128	416	52
000011	3	192	416	52
000100	4	256	416	52
000101	5	320	672	84
000110	6	384	672	84
000111	7	448	672	84
001000	8	512	672	84
001001	9	576	928	116
001010	10	640	928	116
001011	11	704	928	116
001100	12	768	928	116
001101	13	832	1184	148
001110	14	896	1184	148
001111	15	960	1184	148
010000	16	1024	1184	148
...	...	...		
100000	32	2048	2208	276
...	...	...		
110000	48	3072	3232	404
...	...	...		
000000	64	4096	4256	532

48.11.7.4 Key indexing

With the KEYIDX bits a key can be assigned a specific index value. It is done during the Set Key and Generate Key operations. The value of KEYIDX is part of the Key Code.

During key reconstruction the index defined in the Key Code is output on the KEYOUTIDX bits in the KEYOUTINDEX register. It can be used to send the key to a specific target.

Keys with key index 0 are sent to the AES or PRINCE key interface. Keys with other indexes are sent to the key register KEYOUTPUT.

Example:

Assume a key is intended to be used by a software AES encryption module that has number 0xA assigned to it. Before the Set Key operation the KEYIDX bits in the KEYINDEX register are set to 0xA. Then the Set Key operation is started and the key is passed to the PUF. The resulting Key Code (KC) is stored.

When the key is required, the Get Key operation is started and the KC is passed to PUF. The key index related to this KC appears on KEYOUTIDX in the KEYOUTINDEX register and the resulting key appears in the KEYOUTPUT register. Using the KEYOUTIDX value the key can be sent to the AES target with number 0xA.

48.11.7.5 Key code header

The first 32 bits of the Key Code comprise a header with information about the type of key it represents. It includes three fields. See Table 993. The unused bits are always 0.

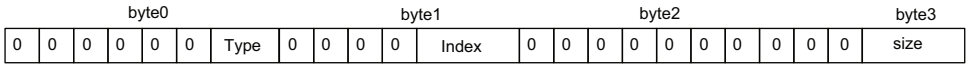


Fig 186. KC header format

Table 993. KC header field description

Name	Description
Type	Define key type. 11: Reserved. 10: Reserved. 01: Generate key. 00: User key.
Index	Value of KEYIDX at the moment of the Set Key or Generate Key operation.
Size	Value of KEYSIZE at the moment of the Set Key or Generate Key command.

48.11.7.6 Key byte order on the APB interface

The first word contains the first bytes and the second word the next bytes. Figure 187 shows the byte 16 order within a word.

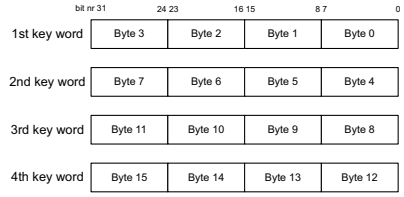


Fig 187. Key byte order on the Key interface for 128-bit key

48.11.7.7 Enroll

During the enroll operation the SRAM startup values are read. Based on these a device specific digital fingerprint is derived and the related activation code (AC) is generated. The following steps are performed:

1. Software gives the enroll command to PUF by setting ENROLL bit.
2. PUF reads the startup values and makes the AC available through CODEOUTPUT.
3. After the operation is finished the PUF de-asserts BUSY and asserts SUCCESS, signaling that the enroll operation has completed successfully.

48.11.7.8 Start

During the start operation the SRAM startup values and the activation code are read. Based on these the PUF reconstructs the digital fingerprint. The following steps are performed:

1. Client Design gives the Start command to PUF through START.
2. PUF reads the startup values and requests for the AC through CODEINPUT.
3. After the operation is finished the PUF de-asserts BUSY and asserts SUCCESS, signaling that the Start operation has completed successfully. When ERROR is asserted, instead of SUCCESS, the provided activation code does not match the device. In this case the PUF as gone to the Error state. See Figure 185 and Section 48.11.7.13 “Error response”.

48.11.7.9 Generate key

During the Generate key operation, the key size and key index are defined first and a device-specific key is generated. Based on this the device-specific Key Code (KC) is generated. The following steps are performed:

1. Software sets KEYIDX and KEYSIZE to their required values.
2. Software gives the Generate Key command to the PUF controller using GENERATEKEY.
3. The PUF controller makes the KC available through CODEOUTPUT.
4. After the operation is finished, the PUF controller de-asserts BUSY and asserts SUCCESS, signaling that the Generate key operation has completed successfully.

48.11.7.10 Set key

During this operation, the key size and key index are defined first the user key is read. Based on this the device-specific Key Code (KC) is generated. The following steps are performed:

1. Set KEYIDX and KEYSIZE to their required values.
2. Give the Set User Key command to the PUF controller using SETKEY.
3. Issue the request for the user key using the KEYINPUT.
4. The KC is available through CODEOUTPUT.
5. After the operation is finished, the PUF controller de-asserts BUSY and asserts SUCCESS, signaling that the Set Key operation has completed successfully.

Remark: Keys with key index 0 are sent to the AES PRINCE Key interface. When user key index is 0, write user key to KEYINPUT register Least Significant word first. User Keys with other indexes should be written to the KEYINPUT register with Most Significant word first.

48.11.7.11 Get Key

During Get Key operation the key code is read. The key code includes the key index and key size; the values for KEYIDX an KEYSIZE are ignored. The following steps are performed:

1. Issue the Get Key command using the GETKEY.
2. The key is available using the interface indicated in Table 994. See Section 48.11.7.4 “Key indexing” for more information on using KEYOUTIDX bits.
3. After the operation is finished BUSY is de-asserted and SUCCESS is asserted. It indicates that the Get Key operation has completed successfully. If ERROR is asserted instead of SUCCESS, the provided key code does not match the device. In this case no key is provided and the PUF controller goes to the Error state. See Figure 185 and Section 48.11.7.13 “Error response”.

Note: All key bits produced as defined in the KC must be consumed. If less key bits are consumed as defined in the KC, the PUF controller stays busy until the remaining bits are consumed.

Table 994. Key target interfaces per key index

Value of KEYINDEX during set key	Key output on
0	Dedicated interface: KEYINDEX = 0.
Other	PUF key output register.

48.11.7.12 Zeroize

When the ZEROIZE bit is programmed to 1, all internal critical security parameters are erased and the PUF controller goes to the error state. See Figure 185. No new operations can be performed until the device is repowered.

48.11.7.13 Error response

When an error other than an APB error is detected, all internal critical security parameters are erased and the PUF controller goes to the error state.

When the error occurs during a command (for example, when a wrong activation code or key code is given) or during initialization (for example, a reset was given after Zeroize instead of a repower), ERROR is asserted and BUSY is de-asserted, independent of the state of the command signals.

48.11.7.14 Key index blocking

When index blocking register is programmed, key output from blocked index is not possible. However, key code from the blocked index is still readable.

Index 1-7 and Index 8-15 are grouped together. Once index settings are done, the lock should be applied to disable modification until the next system reset.

48.11.8 Software development

This section provides pseudocode drivers that implement the basic functionality to help software development. It is not intended to be optimal code. The code is based on the commands described in Section 48.11.7 “Using PUF”.

The software that interfaces with the PUF controller must read its status and drive the control bits. Also, it must provide input data to the PUF controller and accept output data from the PUF controller.

This section provides high-level code that can be used as a starting point for the development of the driver code. The example code uses status polling to control the flow.

Note: The status polling method is used for clarity. For more efficient operation, an interrupt-driven architecture is recommended.

After reset (with or without a power cycle of the PUF SRAM), the PUF controller is initialized, indicated by busy asserted. The system waits for initialization to be finished before it starts issuing commands. Use the function wait_for_init for this.

The code includes a ZEROIZE function. It can be used in case software detects a reason to delete sensitive data, for example, when an ERROR status is returned by the PUF controller functions.

Table 995 defines the parameters of the functions. Table 996 defines the data access functions; these must be supplied by the system.

Key formats are defined in section Section 48.11.7.6 “Key byte order on the APB interface”. It is assumed that the data and key are stored in memory in this format.

Table 995. Function parameters

Parameter	Description
ACdata	Pointer to a data structure that can store or contains the AC data and the current location in the data. When ACdata is generated it should be stored in some kind of NVM. When ACdata is requested it should be read from NVM.
KCdata	Pointer to a data structure that can store or contains the KC data and the current location in the data. When KCdata is generated it should be stored in some kind of NVM. When KCdata is requested it should be read from NVM.
KeyData	Pointer to a data structure that can store the key data and the current location in the data.
KeySize	Size of the key in bits.
KeyIndex	Index for which the key is targeted.

Table 996. Data access functions

Variable	Description
Initialize(Target)	Empties the target data structure.
Get_data(Data, Source)	Retrieves the next data word from the Source structure, puts it in Data, and removes it from the head.
Append_data(Data, Target)	Appends the data in data to the end of target.

48.11.8.1 Pseudocode wait for Initialization function

```
status wait_for_init() {
    // wait until initialization has finished
    while (*STAT & BUSY != 0) {}
    // check that initialization has passed
    if (*STAT & (SUCCESS | ERROR) != SUCCESS) {
        return ERROR
    }
return OK
}
```

48.11.8.2 Pseudocode enroll function

```
status enroll(ACdata) {
    // clear the ACdata storage
    initialize(ACdata)
    // check if Enroll is allowed
    if (*ALLOW & ALLOWENROLL == 0) {
        return NOT_ALLOWED
    }
    // begin Enroll
    *CTRL = ENROLL
    // wait till command is accepted
    while (*STAT & (BUSY | ERROR) == 0) {
    }
    // while busy read AC
    while (*STAT & BUSY != 0) {
        if (*STAT & CODEOUTAVAIL != 0) {
            tempData = *CODEOUTPUT
            append_data(tempData, ACdata)
        }
    } // while
    // check result
    if (*STAT & SUCCESS == 0) {
        return ERROR
    }
    return OK
}
```

48.11.8.3 Pseudocode start function

```
status start(ACdata) {
    // check if Start is allowed
        if (*ALLOW & ALLOWSTART == 0) {
```

```
        return NOT_ALLOWED
    }
    // begin Start
    *CTRL = START
    // wait till command is accepted
    while (*STAT & (BUSY | ERROR) == 0) {
    }
    // while busy send AC
    while (*STAT & BUSY != 0) {
        if (*STAT & CODEINREQ != 0) {
            get_data(tempData, ACdata)
            *CODEINPUT = tempData
        }
    } // while
    // check result
    if (*STAT & SUCCESS == 0) {
        return ERROR
    }
    return OK
}
```

48.11.8.4 Pseudocode Generate Key function

```
status set_ik(KCdata, KeyIndex, KeySize) {
    // clear the KCdata storage
    initialize(KCdata)
    // check if Set Key is allowed
    if (*ALLOW & ALLOWSETKEY == 0) {
        return NOT_ALLOWED
    }
    // program the key size and index
    *KEYSIZE = KeySize >> 6 // convert to 64-bit blocks
    *KEYINDEX = KeyIndex
    // begin Set Key
    *CTRL = GENERATEKEY
    // wait till command is accepted
    while (*STAT & (BUSY | ERROR) == 0) {
    }
    // while busy read KC
    while (*STAT & BUSY != 0) {
        if (*STAT & CODEOUTAVAIL != 0) {
            tempData = *CODEOUTPUT
            append_data(tempData, KCdata)
        }
    } // while
    // check result
    if (*STAT & SUCCESS == 0) {
        return ERROR
    }
    return OK
}
```

48.11.8.5 Pseudocode Set Key function

```
status set_uk(KCdata, KeyIndex, UKdata) {
    // clear the KCdata storage
    initialize(KCdata)
    // check if Set Key is allowed
    if (*ALLOW & ALLOWSETKEY == 0) {
        return NOT_ALLOWED
    }
    // detect key size
    KeySize = length_in_bits
    // program the key size and index
    *KEYSIZE = KeySize >> 6 // convert to 64-bit blocks
    *KEYINDEX = KeyIndex
    // begin Set Key
    *CTRL = SETUSERKEY
    // wait till command is accepted
    while (*STAT & (BUSY | ERROR) == 0) {}
    // while busy write key and read KC
    while (*STAT & BUSY != 0) {
        if (*STAT & KEYINREQ != 0) {
            get_data(tempData, keyData)
            *KEYINPUT = tempData
        }
        if (*STAT & CODEOUTAVAIL != 0) {
            tempData = *CODEOUTPUT
            append_data(tempData, KCdata)
        }
    } // while
    // check result
    if (*STAT & SUCCESS == 0) {
        return ERROR
    }
    return OK
}
```

48.11.8.6 Pseudocode Get Key function

```
status get_key(KCdata, KeyIndex, KeyData) {
    // clear the KeyData storage
    initialize(KeyData)
    // put unused value in KeyIndex
    // Indicates key transfer via dedicated key interface
    KeyIndex = 255
    // check if Get Key is allowed
    if (*ALLOW & ALLOWGETKEY == 0) {
        return NOT_ALLOWED
    }
    // begin Get Key
    *CTRL = GETKEY
    // wait till command is accepted
    while (*STAT & (BUSY | ERROR) == 0) {
```

```
    }
    // while busy send KC, read key
    while (*STAT & BUSY != 0) {
        if (*STAT & CODEINREQ != 0) {
            get_data(tempData, KCdata)
            *CODEINPUT = tempData
        }
        if (STAT & KEYOUTAVAIL != 0) {
            KeyIndex = *KEYOUTINDEX
            tempData = *KEYOUTPUT)
            append_data(tempData, KeyData)
        }
    } // while
    // check result
    if (*STAT & SUCCESS == 0) {
        return ERROR
    }
    return OK
}
```

48.11.8.7 Pseudocode Zeroize function

```
status zeroize() {
    // zeroize command is always allowed
    *CTRL = ZEROIZE
    // check that command is accepted
    if ((*STAT & ERROR == 0) ||
        (*ALLOW != 0)) {
        return ERROR
    }
    return OK
}
```

48.12 AES engine functional details

The AES engine supports 128-bit, 192-bit, or 256-bit keys for encryption and decryption operations.

48.12.1 Features

- Encryption and decryption of data.
- Secure storage of AES key that cannot be read.
- AES engine peak performance of 0.5 bytes/clock cycle.
- AES engine supports 128-bit, 192-bit or 256-bit key in:
  - Electronic Code Book (ECB) mode.
  - Cipher Block Chaining (CBC) mode.
  - Counter (CTR) mode.
- The AES engine supports 128-bit key in ICB (Indexed Code Book) mode, that offers protection against side-channel attacks.

- The AES engine is compliant with the FIPS (Federal Information Processing Standard) Publication 197, Advanced Encryption Standard (AES).
- AES offers programmability to select little-endian or big-endian mode of operation.
- It may use the processor, DMA or AHB Master for data movement. AHB Master may only be used to load data, DMA may be used to read-out results. DMA based result reading is a *trigger*, so the application must set the size correctly.

48.12.2 Basic configuration

- AES functionality is combined with SHA block, referred to as SHA-AES. For clock and reset connection programmability information please refer to SHA basic configuration section Section 48.14.3 “Status register”.
- For AES registers please refer to SHA-AES register description Section 48.16 “Register description”.
- AES block shares same base address as SHA block.

48.12.3 General description

The AES being a block cipher, encrypts and decrypts the user provided 128-bit block data Electronic Code Book (ECB) model. In ECB mode the application provides a 128-bit input block and the AES encrypts or decrypts that into a 128-bit output block. In other two cipher modes supported by this module (CBC, CTR) the application first provides an IV of 128-bits, and then provides 128-bit data blocks. The peripheral will XOR the data with the appropriate component and then process the next.

The AES engine has an output/digest buffer of 256-bits, and two 512-bit buffers. It uses the first buffer to hold the key of any of three sizes: the IV/nonce of CBC or IV + counter of CTR, and the message block itself. The second buffer may be the output and a cache of the message for CBC decrypt. CBC decrypt needs to XOR in the message at the end, so it is held to allow the next message buffer to be copied in.

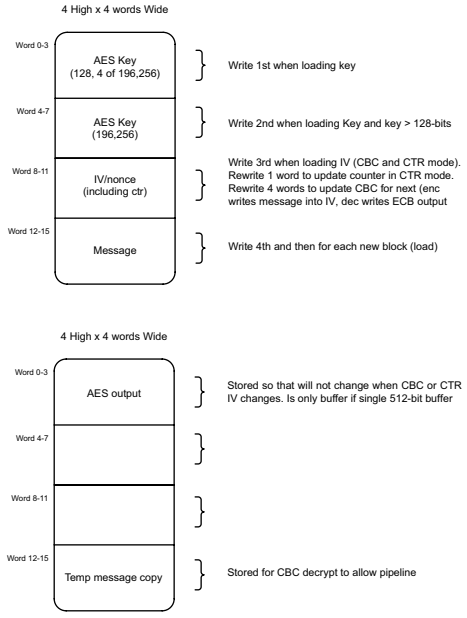


Fig 188. 2x512-bit buffers are used for AES

48.12.4 Using AES engine

1. When starting a new operation, write the CTRL register NEW bit to initialize.
2. The AES engine uses 128-bit, 192-bit or 256-bit key depending on AESKEYSZ filed in CRYPTCFG register. Key can be HW supplied by PUF or supplied by SW
   - If Key is supplied from PUF, see Section 48.2.1 “Key storage/management”, which describes PUF Key Loading for AES.
   - If the key is selected as SW provided, write the 4, 6, or 8 key word values into INDATA. These will be placed in the correct place in buffer1. The STAT register NEEDKEY will be 1 until it is completed.
3. The software defined keys are not retained during power-down or deep-power down and must be reloaded on reset. PUF keys are retrained during power-down but not during deep-power down.
4. If a Cipher mode is selected (rather than just ECB), write the IV/nonce using four words 128-bits. These will be placed in the correct place in buffer1. The STAT register NEEDIV will be 1 until it is completed. It may be the AHB master if enabled.
5. Read in the next 128-bit block of plain text or cipher text.
   - If AHB master is used for read, it will read in the four words.
   - If DMA or processor is used for read, the corresponding one will be notified to provide the four words
6. As soon as the four words are written in, the encryption or decryption starts.
   - If a cipher mode is used, the block being processed will correspond to the rules of that mode.

7. On completion, the data is ready in the OUTDATA0 first four words. The steps allow for load next and the read out of data:
   - If AHB master is used for read and the count is not 0, it will load in the next four words first. It allows the next block to start before the read out of the previous digest, so saving time. The processor or DMA may also do this.
   - If DMA is set for out, it will trigger for reading out the four words. Else, the processor will do it via interrupt.

48.12.5 AES performance

The AES block will take 33+2 cycles for each block to encrypt when using 128-bit keys. Using 192-bit key adds six cycles and 256-bit key adds twelve more cycles.

To decrypt, the AES block will take 43+2 cycles for each block to encrypt when using 128-bit keys. Using 192-bit key adds six cycles and 256-bit key adds twelve more cycles.

The total time required also includes first input for example, optional input of key and IV, as well as input of first four word message data and final output copying final four words. All other input data and output results will be pipe lined and so do not add to the cost unless application or DMA is very slow.

48.12.6 ICB-AES

Whereas AES as a block cipher is an Electronic Code Book, a special hybrid cipher mode is available, called ICB. ICB is a form of CTR cipher mode, but its purpose is to be Side Channel Analysis resistant. It uses a method which is a counter-measure to various Side Channel attacks such as SPA/DPA/DPX (power analysis) and emanation analysis. ICB is slower than normal AES ECB and CTR mode, as a consequence of being SCA resistant.This mode can be used for extra-secure on-chip storage for sensitive information.

In ICB-AES mode first block out would be 32x, 64x, 96x, or 128x slower than normal AES. To make it faster for a stream of blocks, the ICB mechanism will store a partial result into word 4-7 of the second message buffer. It means the second and subsequent blocks are only 3, 4, 5, or 6x slower than a normal AES block, in exchange for being able to run a stream of 8, 16, 32, or 64 blocks before starting a new one.

The MASK (word 8-11) should be randomly generated each time, to remove any leakage from the input of the key or output store of the data

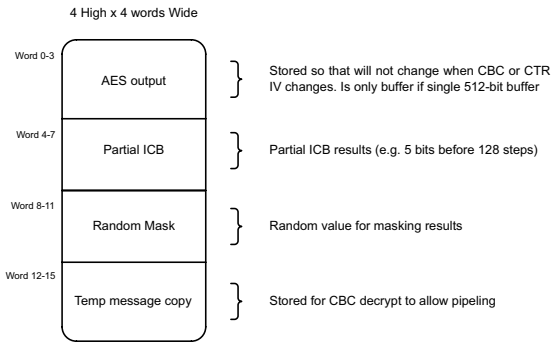


Fig 189. Showing extra values stored in message 2 for ICB

48.12.7 Using Indexed Code Book AES (ICB-AES) engine

To use ICB-AES, a few rules must be followed:

1. The application should write the four word random mask into the MASK register. It can be written in any order.
2. The application must use little endian for keys and the like.
3. The application may use the secret key; if writing the key, it must mask it with the inverse of the random mask before writing. The block will read out using the same inverse of the random mask as used for writing.
4. The CTR must be aligned to the ICBSZ field of the CRYPTCFG. Therefore, if IV is 64-bits, then set AESCTRPOS value to 4. If IV is 128-bits, set AESCTRPOS value to 0. If IV is 32-bits, then set AESCTRPOS value to 6.
5. The CTR can be written with any starting value wanted, but note that the ICBSTRM model is to say how many bits are in play before it has to start over. So, if not starting with 0 in those bits (the bottom 3, 4, 5, or 6 bits), then the application should stop and reload before the implicit count of 8, 16, 32, or 64.
6. The CRYPTCFG setting must be ECB and encrypt.
7. The data is written masked using the same random mask as written in MASK registers. it means plaintext is never exposed just as keys are never exposed. The formula then is:
   - The decrypted plain text should be unmasked and it should be done carefully to not leak.
   - The ciphered text output of ICB-AES should be unmasked prior to storing

48.12.8 ICB-AES performance

ICB-AES trades off performance for SCA (Side Channel Analysis) countermeasure protection. The speed for the first block using a new IV+ctr is 32x, 64x, 96x, or 128x slower than one AES ECB operation.

48.13 HASH functional details

48.13.1 Features

- Performs SHA-1 and SHA-2(256) based hashing.
- Used with HMAC to support a challenge/response or to validate a message.

48.13.2 Basic configuration

- The priority for the SHA engine can be set in PRI_SHA bit in AHBMATPRIO register. See Table 40.
- The SHA engine can be reset using the SHA_RST bit in PRESETCTRL2 register. See Table 47.
- The clock to the SHA engine can be controlled using the SHA bit in AHBCLKCTRL2 register. See Table 57.

48.13.3 General description

The SHA engine processes blocks of 512-bits (16 words) at a time and performs the SHA-1 hashing in 80 clock cycles per block or SHA-256 hashing in 64 clock cycles per block. As many blocks as needed may be processed. The last block must be formatted per the SHA model:

1. The last data must be 447 bits or less. If more, then an extra block must be created.
2. After the last bit of data, a '1' bit is appended. Then, as many 0 bits are appended to take it to 448 bits long (so, 0 or more).
3. Finally, the last 64-bits contain the length of the whole message, in bits, formatted as a word.

For example, if a message is an exact multiple of 512-bits, create an extra block. The first bit of the last block will be a 1 followed by 447 zeroes. The remaining 64-bits will contain the length of the whole message including the last block.

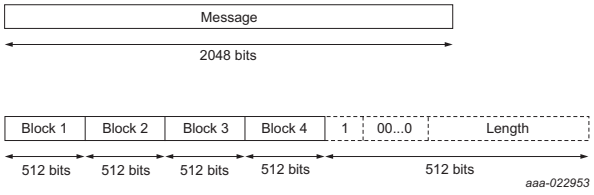


Fig 190. SHA input data block

The Arm processor uses little-endian and therefore, the SHA engine reverses the bytes in the words written to the data register to big-endian format. It is because a hash is on bytes, so a string such as "abcd", when read as a word by the processor (or DMA) is reversed into "dcba". When the input data is provided in little-endian format, the hash block swaps them to process correctly.

48.13.4 Security lock and register access

If a security level has locked the block using the LOCK register, no register is readable or writable from a lower level, except the LOCK register can be read any time as well as the ID at offset 0xFFC.

48.14 HASH-AES Register description

Table 997. Register overview: (HASH-AES, base address = 0x400A 4000)

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	Control register.	0x0	48.14.2
STATUS	R/W1C	0x004	Status register.	0x0	48.14.3
INTENSET	R/W	0x008	Interrupt enable register.	0x0	48.14.4
INTENCLR	W1C	0x00C	Interrupt clear register.	-	48.14.5
MEMCTRL	R/W	0x010	Memory control register.	0x0	48.14.6
MEMADDR	R/W	0x014	Memory address register.	0x0	48.14.7
INDATA	WO	0x020	Input data register.	0x0	48.14.8
ALIAS0	WO	0x024	Alias0 register.	-	48.14.8
ALIAS1	WO	0x28	Alias1 register.	-	48.14.8
ALIAS2	WO	0x2C	Alias2 register.	-	48.14.8
ALIAS3	WO	0x30	Alias3 register.	-	48.14.8
ALIAS4	WO	0x34	Alias4 register.	-	48.14.8
ALIAS5	WO	0x38	Alias5 register.	-	48.14.8
ALIAS6	WO	0x3C	Alias6 register.	-	48.14.8
DIGEST0/OUTDATA0	RO	0x040	Digest 0 register.	0x0	48.14.9
DIGEST1/OUTDATA	RO	0x44	Digest 1 register.	0x0	48.14.9
DIGEST2/OUTDATA0	RO	0x48	Digest 2 register.	0x0	48.14.9
DIGEST3/OUTDATA0	RO	0x4C	Digest 3 register.	0x0	48.14.9
DIGEST4/OUTDATA0	RO	0x50	Digest 4 register.	0x0	48.14.9
DIGEST5/OUTDATA0	RO	0x54	Digest 5 register.	0x0	48.14.9
DIGEST6/OUTDATA0	RO	0x58	Digest 6 register.	0x0	48.14.9
DIGEST7/OUTDATA0	RO	0x5C	Digest 7 register.	0x0	48.14.9
CRYPTCFG	RW	0x80	CRYPTCFG.	0x0	48.14.10
CONFIG	RO	0x84	CONFIG.	0x9CB	48.14.11
LOCK	RW	0x8C	LOCK.	0x0	48.14.12
MASK0	WO	0x90	MASK0.	0x0	48.14.13
MASK1	WO	0x94	MASK1.	0x0	48.14.13
MASK2	WO	0x98	MASK2.	0x0	48.14.13
MASK3	WO	0x9C	MASK3.	0x0	48.14.13

48.14.1 Usage

Following section describes programming sequence for RNG module and relevant system settings for few RNG use-scenarios.

48.14.2 Control register

The control register is used to configure the HASH-AES engine. The HASH-AES engine is enabled when the MODE bit is selected to SHA-1 or SHA-256. The NEW bit field is written to 1, before the data can be loaded into INDATA (or its aliases, or both) register.

Table 998. Control register (CTRL, offset = 0x000)

Bit	Symbol	Value	Description	Reset value
2:0	MODE		This field is used to select the operational mode of SHA engine.	0x0
		0x0	Disabled.	
		0x1	SHA-1 is enabled.	
		0x2	SHA-256 is enabled.	
		0x3	Reserved.	
		0x4	AES	
		0x5	ICB-AES	
		0x6	Reserved	
		0x7	Reserved	
3	-	-	Reserved.	-
4	NEW_HASH		When this bit is set, a new hash operation is started. It automatically self-clears in one clock cycle. **Remark:** The WAITING bit in Status register gets cleared for one cycle during initialization.	0x0
7:5	-	-	Reserved.	-
8	DMA_I		Written with 1 to use DMA to fill INDATA. If Hash, will request from DMA for 16 words and then will process the Hash. If Cryptographic, it will load as many words as needed, including key if not already loaded. It will then request again. Normal model is that the DMA interrupts the processor when its length expires. Note that if the processor will write the key and optionally IV, it should not enable this until it has done so. Otherwise, the DMA will be expected to load those for the 1st block (when needed).	0x0
		0	DMA disabled.	
		1	DMA enabled.	
9	DMA_O		Written to 1 to use DMA to drain the digest/output. If both DMA_I and DMA_O are set, the DMA has to know to switch direction and the locations. This can be used for crypto uses.	0x0
		0	DMA disabled.	
		1	DMA enabled.	
11:10	-	-	Reserved.	-
12	HASHSWPB		If 1, will swap bytes in the word for SHA hashing. The default is byte order (so, LSB is 1st byte) but this allows swapping to MSB is first such as is shown in SHS spec. For cryptographic swapping, see the CRYPTCFG register.	0x0
31:13	-	-	Reserved.	-

48.14.3 Status register

The Status register indicates the status of the Hash-AES peripheral. It shows when the SHA engine is waiting for data and when the results are available. These bits correspond to both interrupts and DMA (in the case of data).

Table 999. Status register (STATUS, offset = 0x4)

Bit	Symbol	Access	Value	Description	Reset value
0	WAITING	RO		If 1, the block is waiting for more data to process.	0x0
			0	Not waiting for data, SHA may be disabled or may be busy. Note: That for cryptographic uses, it is not set if IsLast is set nor will it set until at least 1 word is read of the output.	
			1	Waiting for data to be written in (16 words).	
1	DIGEST	RO		For Hash, if 1 then a DIGEST is ready and waiting and there is no active next block already started. For Cryptographic uses, this will be set for each block processed, indicating OUTDATA (and OUTDATA2 if larger output) contains the next value to read out. It is cleared when any data is written, when New is written, for Cryptographic uses when the last word is read out, or when the block is disabled.	0x0
			0	No digest is ready.	
			1	Digest is ready. Application may read it or may write more data.	
2	ERROR	W1C		If 1, an error occurred. For normal uses, this is due to an attempted overrun: INDATA was written when it was not appropriate. For Master cases, this is an AHB bus error, the COUNT field will indicate which block it was on.	0x0
			0	No error.	
			1	An error occurred since last cleared (written 1 to clear).	
3		WO		Reserved.	undefined
4	NEEDKEY	RO		Indicates the block wants the key to be written in (set along with WAITING).	0x0
			0	No Key is needed and writes will not be treated as Key.	
			1	Key is needed and INDATA/ALIAS will be accepted as Key. Will also set WAITING.	
5	NEEDIV	RO		Indicates the block wants an IV/NONE to be written in (set along with WAITING).	0x0
			0	No IV/Nonce is needed, either because written already or because not needed.	
			1	IV/Nonce is needed and INDATA/ALIAS will be accepted as IV/Nonce. Will also set WAITING.	
15:6		WO		Reserved.	undefined
21:16	ICBIDX	RO		If ICB-AES is selected, then reads as the ICB index count based on ICBSTRM (from CRYPTCFG). That is, if 3 bits of ICBSTRM, then this will count from 0 to 7 and then back to 0. On 0, it has to compute the full ICB, quicker when not 0.	0x0
31:22		WO		Reserved.	undefined

48.14.4 Interrupt enable register

The Interrupt enable register is used to enable interrupt sources that cause processor interrupts.

Table 1000.Interrupt enable register (INTENSET, offset = 0x00B)

Bit	Symbol	Value	Description	Reset value
0	WAITING		This field indicates if interrupt should be enabled when waiting for input data. The interrupt is cleared when any data is written to INDATA or ALIAS registers.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
1	DIGEST		This field indicates if interrupt is generated when Digest is ready (completed a Hash or completed a full sequence). The interrupt is cleared when any data is written to INDATA or ALIAS registers, when NEW bit is written, or when the SHA engine is disabled.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
2	ERROR		This field indicates if interrupt is generated on an ERROR (as defined in STAT register)	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
31:3	-	-	Reserved.	

48.14.5 Interrupt clear register

The Interrupt clear register is used to clear the interrupt mask enabled by the INTENSET register.

Table 1001.Interrupt clear register (INTENCLR, offset = 0x00C)

Bit	Symbol	Description	Reset value
0	WAITING	Writing a 1 clears the interrupt enabled by the INTENSET register.	0
1	DIGEST	Writing a 1 clears the interrupt enabled by the INTENSET register.	0
2	ERROR	Writing a 1 clears the interrupt enabled by the INTENSET register.	0
31:3	-	Reserved.	-

48.14.6 Memory control register

The Memory Control Register (MEMCTRL) allows setting up the SHA engine to be the AHB bus master to read memory for hashing. It can be used to read 512-bit blocks from SRAM0, and SRAMX. The starting location must be word aligned and the length may be up to 128 kB.

Table 1002.Memory control register (MEMCTRL, offset = 0x010)

Bit	Symbol	Value	Description	Reset value
0	MASTER		This field is used to enable SHA engine as AHB bus master.	0
		0	SHA engine is not AHB bus master and the DMA or Interrupt based model is used with INDATA.	
		1	Enables SHA engine as AHB bus master. DMA and INDATA should not be used.	
15:1	-	-	Reserved.	

Table 1002.Memory control register (MEMCTRL, offset = 0x010) *…continued*

Bit	Symbol	Value	Description	Reset value
26:16	COUNT		This field indicates the number of 512-bit blocks to copy starting at MEMADDR. This register will decrement after each block is copied, ending in 0. The DIGEST interrupt will occur when it reaches 0. If a bus error occurs, it will stop with this field set to indicate the block that failed.	0
		0	Done. Nothing to process.	
		0x1	One 512-bit block to hash.	
		0x2	Two 512-bit blocks to hash.	
		0x3	Three 512-bit blocks to hash. The maximum number of 512-bit blocks that can be processed is 2047 blocks.	
31:27	-	-	Reserved.	

48.14.7 Memory address register

The Memory address register (MEMADDR) holds the base address for MEMCTRL. It must only point to valid locations in SRAM0 or SRAMX, based on the LPC55S6x/LPC55S2x device used and must be word aligned.

Table 1003.Memory address register (MEMADDR, offset = 0x014)

Bit	Symbol	Description	Reset value
31:0	BASE	Address base to start copying from, word aligned (so bits 1:0 must be zero). This field will advance as it processes the words. If it fails with a bus error, the register will contain the failing word.	0

48.14.8 Input data and ALIAS registers

The INDATA and its ALIAS registers are used for writing the 16 words per hash. The aliases exist so the processor can use Store Multiple (STM). The DMA only writes to INDATA.

Table 1004.Input data register (INDATA, offset = 0x020)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

Table 1005.Alias 0 register (ALIAS0, offset = 0x024)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

Table 1006.Alias 1 register (ALIAS1, offset = 0x028)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

Table 1007.Alias 2 register (ALIAS2, offset = 0x02C)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

Table 1008.Alias 3 register (ALIAS3, offset = 0x030)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

Table 1009.Alias 4 register (ALIAS4, offset = 0x034)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

Table 1010.Alias 5 register (ALIAS5, offset = 0x038)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

Table 1011.Alias 6register (ALIAS6, offset = 0x03C)

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA engine swaps the word to the required big-endian format.	0

48.14.9 DIGEST (or OUTDATA0) register

The DIGEST or OUTPUT registers contain the 128-bits, 160-bits or 256-bits, depending on AES, SHA1, SHA256 or crypto or SHA512. The registers are written in word format, therefore, endianness should be considered when sending or comparing. The first five DIGEST registers are populated for SHA-1and all eight DIGEST registers are populated for SHA-256. If SHA-1 is used, DIGEST [0:4] are populated and if SHA-256 is used, DIGEST [0:7] are populated.

Table 1012.DIGEST 0 register (DIGEST0, offset = 0x040)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 1013.DIGEST 1 register (DIGEST1, offset = 0x044)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 1014.DIGEST 2 register (DIGEST2, offset = 0x048)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 1015.DIGEST 3 register (DIGEST3, offset = 0x04C)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 1016.DIGEST 4 register (DIGEST4, offset = 0x050)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 1017.DIGEST 5 register (DIGEST5, offset = 0x054)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 1018.DIGEST 6 register (DIGEST6, offset = 0x058)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 1019.DIGEST 7 register (DIGEST7, offset = 0x05C)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

48.14.10 Cryptographic configuration register

The CRYPTCFG register is the cryptographic configuration register for AES. It is ignored if SHA hashing is selected. Only the fields for the selected encryption scheme will be used and any field relating to a feature not supported will be ignored (writes will have no effect and the bits will read back as 0).

Table 1020. CRYPTCFG register (CRYPTCFG, offset = 0x080)

Bit	Symbol	Value	Description	Reset value
0	MSW1ST_OUT	RW	If 1, OUTDATA0 will be read Most significant word first for AES. Else it will be read in normal little endian - Least significant word first. Note: only if allowed by configuration.	0
1	SWAPKEY	RW	If 1, will SWAP the key input (bytes in each word).	
2	SWAPDAT	RW	If 1, will SWAP the data and IV inputs (bytes in each word).	
3	MSW1ST	RW	If 1, load of key, IV, and data is MSW first for AES. Else, the words are little endian. Note: Only if allowed by configuration.	
5:4	AESMODE	RW	AES Cipher mode to use if plain AES.	0
		0	ECB – used as is.	
		1	CBC mode. See Section 48.12.3 “General description”.	
		2	CTR mode. See Section 48.12.3 “General description”. See AESCTRPOS.	
		3	Reserved.	
6	AESDECRYPT	RW	AES ECB direction. Only encryption used if CTR mode or manual modes such as CFB.	0
		0	Encrypt.	
		1	Decrypt.	
7	AESSECRET	RW	Selects the Hidden Secret key vs. User key, if provided.	
		0	User key provided in normal way.	
		1	Secret key provided in hidden way by HW.	
9:8	AESKEYSZ	RW	Sets the AES key size.	0
		0	128 bit key.	
		1	192 bit key.	
		2	256 bit key.	
		3	Reserved.	
12:10	AESCTRPOS	RW	Half word position of 16b counter in IV if AESMODE is CTR. Only supports 16b counter, so application must control any additional bytes if using more.The 16-bit counter is read from the IV and incremented by 1 each time. Any other use CTR should use ECB directly and do its own XOR and so on.	0
15:11	-	-	Reserved.	-
16	STREAMLAST	RW	Is 1 if last stream block. If not 1, then the engine will compute the next “hash”.	0
19:17	-	-	Reserved.	-
21:20	ICBSZ	RW	This sets the ICB size between 32 bits and 128 bits, using the following rules. Note that the counter is assumed to occupy the low order bits of the IV	0
		0	32 bits of the IV/ctr are used (from 127:96).	
		1	64 bits of the IV/ctr are used (from 127:64).	
		2	96 bits of the IV/ctr are used (from 127:32).	
		3	128 bits of the IV/ctr are used.	

Table 1020. CRYPTCFG register (CRYPTCFG, offset = 0x080) *…continued*

Bit	Symbol	Value	Description	Reset value
23:22	ICBSTRM	RW	The size of the ICB-AES stream that can be pushed before needing to compute a new IV/ctr (counter start). It optimizes the performance of the stream of blocks after the first.	0
		0	Maximum stream of 8 blocks.	
		1	Maximum stream of 16 blocks.	
		2	Maximum stream of 32 blocks.	
		3	Maximum stream of 64 blocks.	
31:24	-	-	Reserved.	-

48.14.11 Configuration register

The Read-Only CONFIG register indicates what features are available in this block. SHA1 and SHA2-256 are always available, so it indicates features beyond that.

Table 1021. CONFIG register (CONFIG, offset = 0x084)

Bit	Symbol	Value	Description	Reset value
0	DUAL	RO	1 if 2 x 512 bit buffers, 0 if only 1 x 512 bit.	0
1	DMA	RO	1 if DMA is connected.	
2	-	-	Reserved.	
3	AHB	RO	1 if AHB Master is enabled.	
5:4	-	-	Reserved.	
6	AES	RO	1 if AES 128 included.	0
7	AESKEY	RO	1 if AES 192 A and 256 also included.	0
8	SECRET	RO	1 if AES Secret key available.	
9		RO	Reserved.	
10		RO	Reserved.	
11	ICB	RO	1 if ICB over AES included.	
31:12	-	-	Reserved.	

48.14.12 LOCK register

The Lock register is used to secure-lock the block from use by lower security levels. When the lock is written, it records the current security level. Only that level and higher may use the block (read or write) until it is unlocked. It may only be unlocked by a security level which is the lock-level or higher. The lock state is readable by any security level along with the ID, but all other registers are masked off to lower levels when locked.

Changing the SECLOCK field clears the user key. However, if secret key is used, it does not get reset by changing the SECLOCK field. The application can reset it using the key reset register. See Section 48.11.6.19 “Key reset register”.

Table 1022. LOCK register (LOCK, offset = 0x80C)

Bit	Symbol	Value	Description	Reset value
1:0	SECLOCK	W, RO	Write 1 to secure-lock this block (if running in a security state). Write 0 to unlock. If locked already, may only write if at same o. higher security level as lock.	0
			Reads as: 0 if unlocked, else 1, 2, 3 to indicate the security level at which it is locked. NOTE: this and ID are the only readable registers if locked and current state is lower than lock level.	
		1	Locks to the current security level. AHB Master will issue requests at this level.	
		0	Unlocks, so block is open to all. But, AHB Master will only issue non-secure requests.	
3:2	-	-	Reserved.	
15:4	PATTERN	RW	Must write 0A75 to change lock state.	
		A75	Pattern needed to change bits 1:0.	
31:16	-	-	Reserved.	

48.14.13 Mask registers

The WO mask registers are written with a random mask to form 128 bits of randomness for masking of the ICB output results. It means that the plaintext is not stored ever, but is always masked.

Table 1023. MASK registers (MASK[0:3], offset [0x090:0x9C])

Bit	Symbol	Value	Description	Reset value
31:0	MASK	WO	A random word.	

48.14.14 Functional description

To perform hashing, select one of the three possible ways to get the input data into the SHA engine:

- Using Cortex-M33 with interrupts:
  - The WAITING and ERROR interrupts are configured in the INTENSET register.
  - When status of the WAITING interrupt in STAT register is 1, the block is loaded by copying the 16 words using INDATA and ALIAS registers.
  - If more blocks need to be loaded, then the WAITING interrupt bit is retained. After the last block is loaded, the DIGEST interrupt is enabled.
- Using the DMA:
  - The DMA is configured for up to 1k words (64-bits, 512-bit blocks) to be read from SRAM0, and SRAMX.Chapter 22 “LPC55S6x/LPC55S2x/LPC552x DMA controller” The SHA peripheral will control the DMA to feed its data as fast as it can. See for configuring the DMA.
  - The SHA engine is double buffered and therefore, allows loading another 16 words while processing the previous words. This pipeline method allows continuous processing of input data.

- An interrupt is used to notify the processor when the DMA transfer is complete. The interrupt service routine can enable the DIGEST interrupt and ERROR interrupt to process the results. Or, it can configure the DMA for more data if needed.
- If the last block is to be constructed separately, then either the DMA can move those 16 words or the processor can do so via interrupts.

- Using AHB Master (when available):
  - The SHA peripheral is enabled as AHB bus master. The memory location to read from SRAM or flash and the number of blocks to read is configured using the MEMCTRL register.
  - The DIGEST and ERROR interrupts are enabled in INTENSET register. The interrupts will not occur until the last block is completed and the digest is computed.
  - If the last block is to be constructed separately, then the interrupt service routine may load the constructed last block (or use the DMA) and it will be interrupted when the DIGEST is ready.

48.14.14.1 Performance of SHA engine

The SHA engine contains two message buffers which can be loaded by CPU, DMA or AHB bus master. The performance of the block depends on the memory from where the input data is fetched (Code RAM, system RAM or flash) and activity on the system bus.

48.14.14.1.1 Input data loaded by CPU

The Cortex-M33 core writes 16 words to start Hashing. The block uses INDATA and ALIAS registers to support write operation to contiguous locations. The Hash operation takes 64 or 80 clock cycles based on SHA-1 or SHA-256 Hash algorithm. The processor can load the next 16 words during the time when the Hash operation is being performed on the previous loaded data.

48.14.14.1.2 Input data loaded by DMA

The DMA loads the 16 words based on requests. The Hash operation takes 64 or 80 clock cycles based on SHA-1 or SHA-256 Hash algorithm. The DMA can request and load the next 16 words during the time when the Hash operation is being performed on the previous loaded data.

48.14.14.1.3 Input data loaded by AHB bus master

The AHB bus master loads 16 words from memory. The Hash operation takes 64 or 80 clock cycles based on SHA-1 or SHA-256 Hash algorithm. The AHB master can load the next 16 words during the time when the Hash operation is being performed on the previous loaded data.

48.14.14.2 Initialization

To setup the SHA engine:

1. Take the SHA engine out of reset mode using the HASH0_RST bit, see Section 4.5.9 “Peripheral reset control 2” and enable the clock to SHA peripheral using the HASH0 bit in the AHBCLKCTRL2 register, see Section 4.5.19 “AHB clock control 2”.

Remark: The SHA peripheral only uses the main AHB clock, so no special clocking or scaling is required.

2. Select SHA-1 or SHA-256 mode using the CTRL register.
3. To start a new Hash write 1 to the NEW bit field in CTRL register. This bit automatically self-clears.
4. To input data into the SHA engine, when using:
   - CPU: Write to INTENSET register to enable the WAITING and ERROR interrupts.
   - DMA:
   - Configure the DMA.
   - Enable the DMA interrupt so the application knows when DMA transfer is done.
   - Set the DMA bit in the CTRL register.
5. AHB master:
   - Enable the DIGEST and ERROR interrupts using INTENSET register.
   - Write to the MEMADDR register with the offset in SRAM or flash.
   - Write to the MEMCTRL register to enable the SHA engine as AHB bus master using the MASTER bit and write the number of 512-bit blocks to process in the COUNT field.

48.14.14.3 Interrupt Service Routine (ISR)

48.14.14.3.1 ISR when using CPU

When using CPU to load data into the SHA engine, the algorithm for ISR is

- If the ERROR bit is set in STAT register, there is an issue with the application code since ERROR means overrun.
- If the WAITING bit is set in STAT register, write 16 words into the SHA peripheral. The fastest method is by using structure copy as shown below. If there are no more blocks after this, clear the WAITING interrupt using INTENCLR register and then set DIGEST using INTENSET.
- The fastest copy is usually

```
struct HASH_W {unsigned v[8];} *src, *dst;
src = (struct HASH_W * )memory_to_read_from; //use location in SRAM0 or SRAMX
dst = (struct HASH_W * )HASH0_INDATA; // indata and aliases
dst[0] = src[0]; // 1st 8
dst[1] = src[1]; // 2nd 8
```

- If the DIGEST bit is set in STAT register, the DIGEST is ready and so process the Digest register (for example, copy) and clear the interrupt using INTENCLR register.

48.14.14.3.2 ISR when using DMA

When the DMA is used for loading the data into the SHA peripheral, the ISR algorithm is:

- If all the input data are loaded into the SHA engine, enable the DIGEST interrupt in the INTENSET register to generate an interrupt when the DIGEST is ready.
- If the last block needs to be manually loaded, write the 16 words now, or use the CPU ISR described in Section 48.14.14.3.1 “ISR when using CPU” to do the one block.

- If the DIGEST bit is set in STAT register, the DIGEST is ready and so process the Digest register (for example, copy) and clear the interrupt using INTENCLR register.

  The ERROR bit is always 0 in this case because an error is not possible.

48.14.14.3.3 ISR for AHB master

The ISR for AHB master is only for DIGEST or ERROR. An ERROR would be a bus error, so the algorithm is:

- If ERROR is set in the STAT register, there is AHB master bus fault. The COUNT field in the MEMCTRL register indicates which block it was processing and the MEMADDR register indicates which memory location it was on when the error occurred.
- If the DIGEST bit is set in STAT register, the DIGEST is ready and so process the Digest register (for example, copy) and clear the interrupt using INTENCLR register.

48.15 RNG functional details

This RNG module is a true random number generator based on two main sources of entropy:

- Phase noise of imprecise clocks derived from the ring oscillators.
- The default values of hundreds of internal flip-flops after a reset.

Other random events based on the availability and behavior of clock cycles are also factored in the process. To remove bias and to whiten the output of TRNG, this block also contains post processing hardware such as Multiply With Carry (MWCG) and Linear Feedback Shift Register (LFSR). This module contains built in test to check the health of the TRNG. User can also read raw entropy directly from physical source.

Note: Please note the functional and operational difference between device revisions 0A and 1B are described in the following sections.

48.15.1 Parameters

The TRNG module relies on the input of four imprecise clocks including a system clock for its random number generation. The output of this TRNG can be used directly in some applications and also to seed and reseed a NIST SP 800-90 defined Deterministic Random Bit Generator (DRBG).

48.15.2 Certification

This TRNG is not certified. However, some advanced checks were done (run test suites, compute statistics on entropy sources) and some stochastic models for entropy sources are available. Guidance for design comes from AIS31 specifications.

48.15.3 Usage

After enabling clocks, the user gets a new random number by reading RANDOM_NUMBER register. Successive random numbers generated by this family of devices starting from power up will pass most test suites including DieHard, NIST SP800-22, FIPS_140-1. This is guaranteed by the quality of the combination of LFSR and MWCG that is being used.

48.15.4 Entropy accumulation

User can either decide to wait for entropy accumulation, or request new random numbers with no delay, and will still be able to get quality numbers that will pass classical test suites.

First option is the preferred way to use this RNG module. This option checks accumulated entropy using CHI SQAURED Test. Steps required to accumulate and test entropy are described in the Section 48.15.5 “Initialization” and Section 48.15.6 “Normal usage”. The quality of initial entropy is based on states of 404 flip-flops which should be random.

48.15.5 Initialization

To perform the initialization, use the following steps:

1. Enable RNG input clock by clearing power down bit (PDRUNCFG0.PDEN_RNG) and setting AHB RNG clock bit in AHBCLKCTRL.RNG register (AHBCLKCTRLSET2 = 0x00002000). Assert TRNG RESET by setting PRESETCTRL2.RNG_RST bit. Release TRNG Reset by clearing PRESETCTRL2.RNG_RST bit. Set other TRNG registers to the default value.

   Note: When the device wakes up from Power Down mode, the TRNG module reset must be asserted before its use.
2. For 0A, activate CHI computing with ONLINE_TEST_CFG.ACTIVATE = 1. For rev 0A devices with default setting of the (COUNTER_CFG.CLOCK_SEL =0), CHI computing is done on all clocks simultaneously. A CHI computing per clock is not available. (For revision 1B, the recommendation is to perform CHI computing only on one specific unprecise clock by selecting COUNTER_CFG.CLOCK_SEL = 4. This setting is needed to accumulating linear entropy. Set COUNTER_CFG.CLOCK_SEL = 4 to perform CHI SQUARED Test and activate CHI computing with ONLINE_TEST_CFG.ACTIVATE = 1).
3. At power on ONLINE_TEST_VAL.MIN_CHI_SQUARED value is higher than ONLINE_TEST_VAL.MAX_CHI_SQUARED. Wait until ONLINE_TEST_VAL.MIN_CHI_SQUARED decreases and becomes smaller than ONLINE_TEST_VAL.MAX_CHI_SQUARED value.
4. If ONLINE_TEST_VAL.MAX_CHI_SQUARED > 4, program ONLINE_TEST_CFG.ACTIVATE = 0 (to reset), if COUNTER_CFG.SHIFT4X < 7, increment COUNTER_CFG.SHIFT4X then go back to step 2. This will start accumulating entropy. When ONLINE_TEST_VAL.MAX_CHI_SQUARED $\leq$ 4, initialization is now complete. After completion of initialization, follow the below steps to read Random number.

48.15.6 Normal usage

Here are usual steps for setting up the entropy:

Steps for design Rev 0A:

1. Keep clocks and CHI computing active.
2. The following steps are required to read a quality Random number, see pseudo code. For device revision 0A, use this pseudo code:

```
tmpNewRand = RNG->RANDOM_NUMBER;
```

```
     for (int idx=0;idx<32;idx++) { // Asking for 32 bit
entropy refill
    while (RNG->COUNTER_VAL.REFRESH_CNT==0) {}
    tmpNewRand = tmpNewRand ^ (RNG->RANDOM_NUMBER);
    }
// now you have a new random number with 32 true random bits
```

3. Perform online CHI computing check by checking ONLINE_TEST_VAL.MAX_CHI_SQUARED value. Wait till ONLINE_TEST_VAL.MAX_CHI_SQUARED becomes smaller or equal than 4.
4. Go to step 2 to read new random number.

Steps for design Rev 1B:

1. Keep Clocks CHI computing active.
2. Wait for COUNTER_VAL.REFRESH_CNT to become 31 to refill fresh entropy since last reading of a random number.
3. Read new Random number by reading RANDOM_NUMBER register. This will reset COUNTER_VAL.REFRESH_CNT to zero.
4. Perform online CHI computing check by checking ONLINE_TEST_VAL.MAX_CHI_SQUARED value. Wait till ONLINE_TEST_VAL.MAX_CHI_SQUARED becomes smaller or equal than 4.
5. Go to step 2 and read new random number.

48.15.7 Checking the state of initial entropy

In some cases you may want to check the state or quality of the initial entropy. Although there is no built-in hardware mechanism for evaluating the quality of the initial entropy, you can use a software workaround:

- Use the procedure described above to read initial number in order to ensure a minimum entropy accumulation after power-up.
- Store the initial values in non-volatile memory and compute some statistics.

48.15.8 Checking run-time entropy

The following indicators can be used to check run-time entropy:

- total failure: if no clock, no random number will be generated and this will halt the system bus leading to an exception.
- low quality run-time entropy: use the embedded CHI computing to detect very poor quality of entropy source. This check is done in the initialization section.

48.16 Register description

Table 1024 shows the registers and their addresses.

Table 1024. Register overview

Name	Access	Offset	Description	Reset value	Section
RANDOM_NUMBER	R	0x0	This register contains a random 32 bit number which is computed on demand, at each time it is read. Weak cryptographic post-processing is used to maximize throughout.	X [1]	48.16.1
COUNTER_VAL	R	0x8	Counter validation register.	0	48.16.2
COUNTER_CFG	RW	0xC	Counter configuration register.	0	48.16.3
ONLINE_TEST_CFG	RW	0x10	Online test configuration.	0	48.16.4
ONLINE_TEST_VAL	R	0x14	Online test validation.	0	48.16.5
MODULEID	R	0xFFC	Module ID.	0	48.16.6

[1]This register provides random number after reset.

48.16.1 Random number register

This register holds random number generated by the engine.

Table 1025. Random number register (RANDOM_NUMBER, offset = 0x0)

Bit	Symbol	Description	Reset value
31:0	RANDOM_NUMBER	Generated Random Number	X [1]

[1] This register provides random number after reset.

48.16.2 Counter validation register

This register provides RNG relevant information for evaluation and certification purposes.

Table 1026.Counter validation register (COUNTER_VAL, offset = 0X8)

Bit	Symbol	Description	Reset value
7:0	CLK_RATIO	Gives the ratio between the internal clocks frequencies and the register clock frequency for evaluation and certification purposes. The lsb of the CLK_RATIO bit-field provides raw entropy. This entropy can be read each time COUNTER_VAL.REFRESH_CNT increases.	0
12:8	REFRESH_CNT	Incremented (till max possible value) each time COUNTER_VAL.CLK_RATIO is updated. It gives an indication on 'entropy refill' between two reads to RANDOM_NUMBER register (any access to RANDOM_NUMBER register will reset this counter). For Example with COUNTER_CFG.MODE = 2, COUNTER_CFG.CLK_SEL >0, and ONLINE_TEST_CFG.DATA_SEL =0: if 'chi' is correct (ONLINE_TEST_VAL.MAX_CHI_SQUARED becomes smaller than 4), then any increase in REFRESH_CNT gives the indication that at least 1 bit of entropy was generated since last reading to RANDOM_NUMBER register, due to an uncertainty of 1 analog clock cycle. Note: It is not linear accumulation: uncertainty (on number of analog clock periods since last access) increases with square root of the value of COUNTER_VAL.REFRESH_CNT, assuming that incoming analog clocks are variable enough to guarantee independence of two consecutive runs of measurements. Entropy accumulation is linear on revision 1B.	0
31:13	-	Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.	0

48.16.3 COUNTER configuration register

This register provides RNG relevant settings for evaluation and certification purposes. See Section 48.15.2 “Certification”.

Table 1027.Counter configuration register (COUNTER_CFG, offset = 0XC)

Bit	Symbol	Description	Reset value
1:0	MODE	00: Counter disabled. 01: Counter update once. Will return to 00 once done. 10: free running: updates continuously. It activates feature 'enhanced entropy refill', with some spreading among all RNGs.	0
4:2	CLOCK_SEL	Selects the internal clock on which to compute CHI SQUARE statistics. There are four clock sources and each bit in CLOCK_SEL selects a specific clock. CLOCK_SEL = 0 results in an XOR of all four clocks. 000: Use XOR of all clocks for CHI SQUARE COMPUTING 001: Use first clock for CHI SQUARE COMPUTING 010: Use 2nd clock for CHI SQUARE COMPUTING 011: Use third clock for CHI SQUARE COMPUTING 100: Use fourth clock for CHI SQUARE COMPUTING 101-111 reserved	0
7:5	SHIFT4X	To be used to add precision to COUNTER_VAL.CLK_RATIO and determine 'entropy refill'. Supported range is 0..7. Used as well for ONLINE_TEST. See use of this bit-field in the Initialization section.	0
31:8	-	Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.	

48.16.4 Online test configuration register

Table 1028.Online test configuration register (ONLINE_TEST_CFG, offset = 0X10)

Bit	Symbol	Description	Reset value
0	ACTIVATE	0: CHI Computing disabled. 1: CHI Computing activated.	0
2:1	DATA_SEL	Selects source on which to apply online test. This is for CHI SQUARE statistics only. Random number always uses all four clocks. Entropy can be calculated on one clock or on all clocks. 00: LSB of COUNTER: raw data from one or all sources of entropy 01: Reserved 10: Reserved 11: Reserved ONLINE_TEST_CFG.ACTIVATE should be set to 'disabled' before changing this field.	0
31:3	-	Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.	0

48.16.5 Online test validation register

Table 1029.Online test validation register (ONLINE_TEST_VAL, offset = 0X14)

Bit	Symbol	Description	Reset value
3:0	Reserved	Reserved.	0x0
7:4	MIN_CHI_SQUARED	This field is reset when ONLINE_TEST_CFG.ACTIVATE = 0.	0xF
11:8	MAX_CHI_SQUARED	This field is reset when ONLINE_TEST_CFG.ACTIVATE = 0.	0x0
31:12	-	Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.	0

48.16.6 Module ID register

The ID register identifies the type and revision of the RNG module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 1030.Peripheral Identification register (ID, offset = 0XFFC)

Bit	Symbol	Description	Reset value
31:16	ID	Unique module identifier for this RNG block.	0xA0B8
15:12	MAJ_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	3
11:8	MIN_REV	Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	2
7:0	Reserved	Reserved.	0

48.17 PRINCE real-time encryption or decryption details

48.17.1 Functional details

The LPC55S6x/LPC55S2x supports three regions for encryption and decryption, referred to as crypto regions. See Figure 191 below. Each crypto region resides at a 256 kB address boundary within the flash. For the 640 kB flash size in LPC55S6x/LPC55S2x , the first two crypto regions are 256 kB in size, and the third one is 128 kB. Each crypto region is divided into 8 kB sub-regions which can be individually enabled.

Each crypto region has a dedicated key and IV. It allows multiple images to reside in the flash with an independent encryption base. The key is sourced from PUF via an internal hardware interface, without exposing it on the system bus. See Section 48.11.5 “Key management” for more details.

In Figure 191, each of the three crypto regions of flash is shown, including its base address and size. Crypto region 1 is shown in more detail, with its 32 sub-regions shown as a 4x8 grid of blocks. The highlighted sub-regions marked with “c” are “crypto” enabled, meaning they are enabled for both encryption and decryption. This diagram gives an example of how various sub-regions can be configured. Enabled sub-regions are not required to be contiguous. The IV1 and SKey1 shown are the IV and Key used by the PRINCE when encrypting or decrypting the data in the sub-regions of crypto region 1.

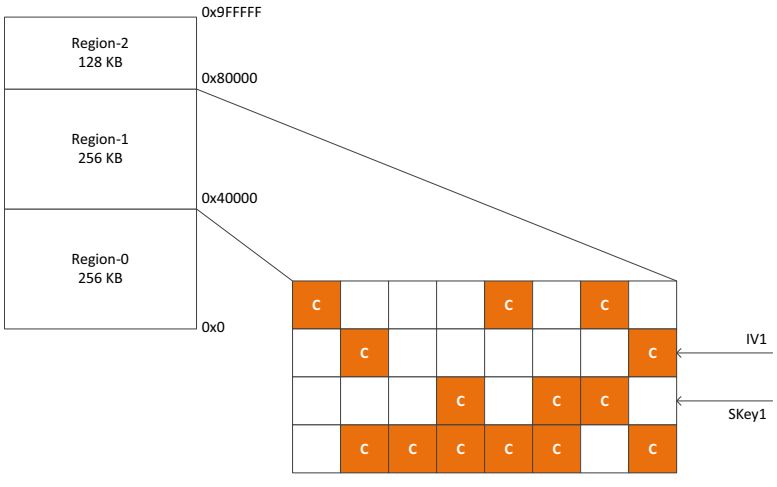


Fig 191. Conceptual diagram of flash crypto regions

PRINCE registers are retained during deep-sleep and power-down but not retained during deep-power down.

48.17.2 Usage notes

During flash programming, the PRINCE control logic monitors the address contained in the STARTA register of the flash controller, see Table 160) to determine when data writes

to DATAW0-DATAW3 registers of the flash controller, see Table 162) are within an enabled sub-region. When they are, the PRINCE encrypts the data written to DATAW0-DATAW3. Therefore, STARTA should always be written *before* DATAW0-DATA3 when programming the flash with encrypted data.

The PRINCE ENC_ENABLE register is provided to control when the PRINCE encrypts the data written to them because the flash controller DATAW0-DATAW3 registers are used for purposes other than flash programming, The PRINCE ENC_ENABLE.EN bit should only be set during flash programming, and cleared for all other flash operations.

To write PRINCE-encrypted data to the flash:

1. Set up the keys for the crypto regions you wish to use as described in Section 48.11.5 "Key management", key loading procedure.
2. Configure the IV, and SR_ENABLE registers as desired, enabling the crypto sub-regions you wish to be encrypted using the PRINCE.
3. Immediately prior to flash programming, set the ENC_ENABLE.EN bit, enabling encryption for sub-regions that have their corresponding SR_ENABLE bit set.
4. Perform flash programming, adhering to the above requirement that STARTA is always written with the flash word address *before* data corresponding to that address is written to the DATAW0-DATAW3 registers.
5. After completion of flash programming clear ENC_ENABLE.EN, to prevent unintended PRINCE encryption of writes to the DATAW0-DATAW3 registers during other flash controller operations.

Remark: The ENC_ENABLE register only controls encryption during flash programming. Data written to a subregion is encrypted when ENC_ENABLE.EN is set, and the corresponding bit for the subregion in SR_ENABLE is set. For flash read data, the SR_ENABLE register exclusively controls whether PRINCE decryption is enabled for a given sub-region.

The MASK and LOCK registers provide added security protection. The value in the MASK registers is used to mask flash data stored in the FMC's data registers. It is a good practice to set this register to a different random value each time the system is booted.

The LOCK register can be used to disable modification of the SR_ENABLE and MASK registers, once they have been set to their desired values.

48.18 PRINCE register descriptions

48.18.1 PRINCE memory map

Table 1031 shows the registers and their addresses.

Table 1031.Register overview: (PRINCE, base address = 5003_5000h)

Name	Access	Offset	Description	Reset value	Section
ENC_ENABLE	RW	0x0	Encryption enable register.		48.18.2
MASK_LSB	WO	0x4	Data mask register, 32 Least Significant Bits.	0000_0000h	48.18.3
MASK_MSB	WO	0x8	Data mask register, 32 Most Significant Bits.	0000_0000h	48.18.4
LOCK	RW	0xC	Lock register.		48.18.5

Table 1031.Register overview: (PRINCE, base address = 5003_5000h) *…continued*

Name	Access	Offset	Description	Reset value	Section
IV_LSB0	WO	0x10	Initial vector register for region 0, Least Significant Bits.	0000_0000h	48.18.6
IV_MSB0	WO	0x14	Initial vector register for region 0, Most Significant Bits.	0000_0000h	48.18.7
BASE_ADDR0	RW	0x18	Base Address for region 0 register.	0000_0000h	48.18.8
SR_ENABLE0	RW	0x1C	Sub-region enable register for region 0	0000_0000h	48.18.9
IV_LSB1	WO	0x20	Initial vector register for region 1, Least Significant Bits.	0000_0000h	48.18.10
IV_MSB1	WO	0x24	Initial vector register for region 1, Most Significant Bits.	0000_0000h	48.18.11
BASE_ADDR1	RW	0x28	Base Address for region 1 register.	0004_0000h	48.18.12
SR_ENABLE1	RW	0x2C	Sub-region enable register for region 1.	0000_0000h	48.18.13
IV_LSB2	WO	0x30	Initial vector register for region 2, Least Significant Bits.	0000_0000h	48.18.14
IV_MSB2	WO	0x34	Initial vector register for region 2, Most Significant Bits.	0000_0000h	48.18.15
BASE_ADDR2	RW	0x38	Base Address for region 2 register.	0008_0000h	48.18.16
SR_ENABLE2	RW	0x3C	Sub-region enable for region 2 register.	0000_0000h	48.18.17

48.18.2 Encryption enable register (ENC_ENABLE)

This register controls whether the PRINCE encryption functionality is enabled.

Table 1032.Encryption enable register (ENC_ENABLE, offset = 0x0)

Bit	Symbol	Access	Value	Description	Reset value
0	EN	RW		Encryption enable.	0x0
			0	Encryption of writes to the flash controller DATAW* registers is disabled.	
			1	Encryption of writes to the flash controller DATAW* registers is controlled by the corresponding SR_ENABLEn register bit.	
31:1		WO		Reserved. Read value is undefined, only zero should be written.	undefined

48.18.3 Data mask register, 32 Least Significant Bits (MASK_LSB)

This register contains the 32 LSBs of the 64-bit data mask applied to data stored in the FMC's data buffers.

Table 1033.Data mask register, 32 Least Significant Bits (MASK_LSB, offset = 0x4)

Bit	Symbol	Access	Value	Description	Reset value
31:0	MASKVAL	WO		Value of the 32 Least Significant Bits of the 64-bit data mask.	0x0

48.18.4 Data mask register, 32 Most Significant Bits (MASK_MSB)

This register contains the 32 MSBs of the 64-bit data mask applied to data stored in the FMC's data buffers.

Table 1034.Data mask register, 32 Most Significant Bits (MASK_MSB, offset = 0x8)

Bit	Symbol	Access	Value	Description	Reset value
31:0	MASKVAL	WO		Value of the 32 Most Significant Bits of the 64-bit data mask.	0x0

48.18.5 Lock register (LOCK)

This register controls whether the SR_ENABLE register of each crypto region is writable. It also controls whether the MASK registers are writable.

Table 1035. Lock register (LOCK, offset = 0xC)

Bit	Symbol	Access	Value	Description	Reset value
0	LOCKREG0	RW		Lock region 0 registers.	0x0
			0	Disabled. SR_ENABLE0 is writable.	
			1	Enabled. SR_ENABLE0 is not writable.	
1	LOCKREG1	RW		Lock region 1 registers.	0x0
			0	Disabled. SR_ENABLE1 is writable.	
			1	Enabled. SR_ENABLE1 is not writable.	
2	LOCKREG2	RW		Lock region 2 registers.	0x0
			0	Disabled. SR_ENABLE2 is writable.	
			1	Enabled. SR_ENABLE2 is not writable.	
7:3		WO		Reserved. Read value is undefined, only zero should be written.	undefined
8	LOCKMASK	RW		Lock the mask registers.	0x0
			0	Disabled. MASK_LSB, and MASK_MSB are writable.	
			1	Enabled. MASK_LSB, and MASK_MSB are not writable.	
31:9		WO		Reserved. Read value is undefined, only zero should be written.	undefined

48.18.6 Initial vector register for region 0, Least Significant Bits (IV_LSB0)

This register contains the 32 LSBs of the PRINCE 64-bit initial vector value for region 0.

Table 1036.Initial vector register for region 0, Least Significant Bits (IV_LSB0, offset = 0x10)

Bit	Symbol	Access	Value	Description	Reset value
31:0	IVVAL	WO		Initial vector value for the 32 Least Significant Bits of the 64-bit initial vector.	0x0

48.18.7 Initial vector register for region 0, Most Significant Bits (IV_MSB0)

This register contains the 32 MSBs of the PRINCE 64-bit initial vector value for region 0.

Table 1037.Initial vector register for region 0, Most Significant Bits (IV_MSB0, offset = 0x14)

Bit	Symbol	Access	Value	Description	Reset value
31:0	IVVAL	WO		Initial vector value for the 32 Most Significant Bits of the 64-bit initial vector.	0x0

48.18.8 Base Address for region 0 register (BASE_ADDR0)

This register provides the base address for region 0, register 0.

Table 1038. Base Address for region 0 register (BASE_ADDR0), offset = 0x18)

Bit	Symbol	Access	Value	Description	Reset value
17:0	ADDR_FIXED	RO		Fixed portion of the base address of region 0.	0x0
19:18	ADDR_PRG	RW		Programmable portion of the base address of region 0.	0x0
31:20	-	-		Reserved.	

48.18.9 Sub-region enable register for region 0 (SR_ENABLE0)

This register enables PRINCE encryption and decryption of data for each sub-region of crypto region 0.

Table 1039.Sub-region enable register for region 0 (SR_ENABLE0, offset = 0x1C)

Bit	Symbol	Access	Value	Description	Reset value
31:0	EN	RW		Each bit in this field enables a sub-region of crypto region 0 at offset 8kB*n, where n is the bit number. A 0 in bit n bit means encryption and decryption of data associated with sub-region n is disabled. A 1 in bit n means that data written to sub-region n during flash programming when ENC_ENABLE.EN = 1 will be encrypted, and flash reads from sub-region n will be decrypted using the PRINCE.	0x0

48.18.10 Initial vector register for region 1, Least Significant Bits (IV_LSB1)

This register contains the 32 LSBs of the PRINCE 64-bit initial vector value for crypto region 1.

Table 1040.Initial vector register for region 1, Least Significant Bits (IV_LSB1, offset = 0x20)

Bit	Symbol	Access	Value	Description	Reset value
31:0	IVVAL	WO		Initial vector value for the 32 Least Significant Bits of the 64-bit Initial Vector.	0x0

48.18.11 Initial vector register for region 1, Most Significant Bits (IV_MSB1)

This register contains the 32 MSBs of the PRINCE 64-bit initial vector value for crypto region 1.

Table 1041.Initial vector register for region 1, Most Significant Bits (IV_MSB1, offset = 0x24)

Bit	Symbol	Access	Value	Description	Reset value
31:0	IVVAL	WO		Initial vector value for the 32 Most Significant Bits of the 64-bit initial vector.	0x0

48.18.12 Base Address for region 1 register (BASE_ADDR1)

This register provides the base address for region 1, register 1. Value is 0x0 after ROM code execution.

Table 1042. Base Address for region 1 register (BASE_ADDR1), offset = 0x28)

Bit	Symbol	Access	Value	Description	Reset value
17:0	ADDR_FIXED	RO		Fixed portion of the base address of region 1.	0x4000
19:18	ADDR_PRG	RW		Programmable portion of the base address of region 1.	0x1
31:20	-	-		Reserved.	

48.18.13 Sub-region enable register for region 1 (SR_ENABLE1)

This register enables PRINCE encryption and decryption of data for each sub-region of crypto region 1.

Table 1043.Sub-region enable register for region 1 (SR_ENABLE1, offset = 0x2C)

Bit	Symbol	Access	Value	Description	Reset value
31:0	EN	RW		Each bit in this field enables a sub-region of crypto region 1 at offset 8kB*n, where n is the bit number. A 0 in bit n bit means encryption and decryption of data associated with sub-region n is disabled. A 1 in bit n means that data written to sub-region n during flash programming when ENC_ENABLE.EN = 1 will be encrypted, and flash reads from sub-region n will be decrypted using the PRINCE.	0x0

48.18.14 Initial vector register for region 2, Least Significant Bits (IV_LSB2)

This register contains the 32 LSBs of the PRINCE 64-bit initial vector value for crypto region 2.

Table 1044.Initial vector register for region 2, Least Significant Bits (IV_LSB2, offset = 0x30)

Bit	Symbol	Access	Value	Description	Reset value
31:0	IVVAL	WO		Initial vector value for the 32 Least Significant Bits of the 64-bit initial vector.	0x0

48.18.15 Initial vector register for region 2, Most Significant Bits (IV_MSB2)

This register contains the 32 MSBs of the PRINCE 64-bit initial vector value for crypto region 2.

Table 1045.Initial vector register for region 2, Most Significant Bits (IV_MSB2, offset = 0x34)

Bit	Symbol	Access	Value	Description	Reset value
31:0	IVVAL	WO		Initial vector value for the 32 Most Significant Bits of the 64-bit initial vector.	0x0

48.18.16 Base Address for region 2 register (BASE_ADDR2)

This register provides the base address for region 2, register 2. Value is 0x0 after ROM code execution.

Table 1046. Base Address for region 2 register (BASE_ADDR2), offset = 0x38)

Bit	Symbol	Access	Value	Description	Reset value
17:0	ADDR_FIXED	RO		Fixed portion of the base address of region 2.	0x8000
19:18	ADDR_PRG	RW		Programmable portion of the base address of region 2.	0x2
31:20	-	-		Reserved.	

48.18.17 Sub-Region enable for region 2 register (SR_ENABLE2)

This register enables PRINCE encryption and decryption of data for each sub-region of crypto region 2.

Table 1047.Sub-region enable register for region 2 (SR_ENABLE2, offset = 0x3C)

Bit	Symbol	Access	Value	Description	Reset value
31:0	EN	RW		Each bit in this field enables a sub-region of crypto region 2 at offset 8 kB*n, where n is the bit number. A 0 in bit n bit means encryption and decryption of data associated with sub-region n is disabled. A 1 in bit n means that data written to sub-region n during flash programming when ENC_ENABLE.EN = 1 will be encrypted, and flash reads from sub-region n will be decrypted using PRINCE.	0x0

49.1 How to read this chapter

This chapter applies to only to the LPC55S6x part.

49.2 Features

The key features of the PowerQuad are:

- 4x single precision floating point MAC.
- AHB DMA - read/write data for input/computations/results. The PowerQuad handles 128-bit wide RAM for input/computations/results.
- Coprocessor interface for tightly coupled opcodes (use two MACs, can run two in parallel): sin(x), cos(x), ln(x), e^(x), e^(-x),1/(x),1/sqrt(x), sqrt(x), biquad(x).
- FFT/iFFT/DCT/iDCT machine.
- Matrix operation: Add, Sub, Dot, Prod, Mult, Inverse, Transpose, Scale.
- Convolution/Correlation/FIR.
- Arctan / Arctanh (Can be customized to compute any CORDIC function).

With the assistance of the PowerQuad, the Cortex-M33 can be freed to perform other tasks. While the PowerQuad is executing the assigned computation task, the CM33 can prepare the next PowerQuad task, resulting in a pipeline of PowerQuad tasks.

49.3 General description

The PowerQuad is a hardware accelerator targeting common calculations in DSP applications. It consists of seven internal computation engines:

- Transform
- Transcendental function
- Trigonometry function
- Dual biquad IIR filter
- Matrix accelerator
- FIR filter
- CORDIC

49.4 Using the PowerQuad with the Cortex-M33

In a conventional MCU without PowerQuad, the main CPU performs the computation work. When a more critical task needs to be serviced, the CM33 has to save registers in the stack, service the task, then retrieve registers from the stack and continue computation work. Using the PowerQuad overcomes the single task flow.

While the conventional approach is to program compute tasks sequentially and anticipate conditional branching, the user is encouraged to treat the PowerQuad as a coprocessor, allowing multiple tasks by rearranging them and lessen the branching to speed up computational processing.

The PowerQuad engines serve to compute the assigned functions that can be up to 25 times faster than the CM33. Here is a performance comparison table for the various functions:

Table 1048.FFT/iFFT/DCT/iDCT functions

Function	Using CM33 (ms)	Using PowerQuad (ms)	Improved Performance	Conditions
rfft_q15	305.9E-6	20.1E-6	15.2	Frequency: 150MHz IAR 8.50.1 SDK 2.9.0 Running on RAM Optimization O3 Size
rfft_q31	380.1E-6	20.1E-6	19.0	
cfft_q15	483.1E-6	22.2E-6	21.7	
cfft_q31	597.5E-6	22.2E-6	26.9	
ifft_q15	481.4E-6	22.2E-6	21.7	
ifft_q31	607.5E-6	22.2E-6	27.4	

Table 1049.Convolution/Correlation/FIR functions

Function	Using CM33 (ms)	Using PowerQuad (ms)	Improved Performance	Conditions
fir_q15	16.2E-6	1.6E-6	10.2	Frequency: 150MHz IAR 8.50.1 SDK 2.9.0 Running on RAM Optimization O3 Size
fir_q31	12.5E-6	1.6E-6	7.8	
fir_f32	15.9E-6	1.6E-6	9.9	
conv_q15	3.0E-6	1.2E-6	2.6	
conv_q31	2.5E-6	1.2E-6	2.2	
conv_f32	2.7E-6	1.2E-6	2.3	
correl_q15	3.2E-6	1.2E-6	2.8	
correl_q31	2.7E-6	1.2E-6	2.3	
correl_f32	2.8E-6	1.2E-6	2.4	

Table 1050.Matrix Engine

Function	Using CM33 (ms)	Using PowerQuad (ms)	Improved Performance	Conditions
mat_add_q15	15.5E-6	5.9E-6	2.6	Frequency: 150MHz IAR 8.50.1 SDK 2.9.0 Running on RAM Optimization O3 Size
mat_add_q31	15.5E-6	5.9E-6	2.6	
mat_add_f32	24.0E-6	5.9E-6	4.0	
mat_sub_q15	15.5E-6	5.9E-6	2.6	
mat_sub_q31	15.5E-6	5.9E-6	2.6	
mat_sub_f32	24.0E-6	5.9E-6	4.0	
mat_mult_q15	212.1E-6	19.1E-6	11.1	
mat_mult_q31	296.7E-6	19.1E-6	15.6	
mat_mult_f32	292.7E-6	43.5E-6	6.7	
mat_inverse_f32	140.0E-6	34.7E-6	4.0	
mat_trans_q15	12.7E-6	4.9E-6	2.6	
mat_trans_q31	12.6E-6	4.9E-6	2.6	
mat_trans_f32	12.7E-6	5.9E-6	2.2	
mat_scale_q15	18.9E-6	5.5E-6	3.4	
mat_scale_q31	20.7E-6	4.9E-6	4.3	
mat_scale_f32	17.2E-6	4.2E-6	4.1	

49.5 PowerQuad operation

After the clock to the PowerQuad is applied, based on the requested operation, one of the computation engines is activated. When the result is ready, an event/interrupt will be generated to inform the CM33 that the result is ready. Alternatively the CM33 can also poll the PowerQuad status.

All PowerQuad operations fall into one of two access types: via the coprocessor interface (CP), and via the AHB slave interface (AHB).

- When accessing the PowerQuad via the coprocessor interface, executing a correctly formatted coprocessor instruction launches it, and executing a second coprocessor instruction retrieves the result.
- When accessing the PowerQuad via the AHB slave interface, writing to the PowerQuad launches it. Completion can optionally generate an interrupt, or be polled via the INST_BUSY bit (cleared to '0') in the CONTROL register. Results are then retrieved via one or more AHB reads.
- In the LPC55S6x, the PowerQuad uses SRAM Bank 4 (16kB at 0x2004 0000 to 3FFF or 0x3004 0000 to 3FFF depending on the Secured or Non Secured Configuration) for scratch pad.

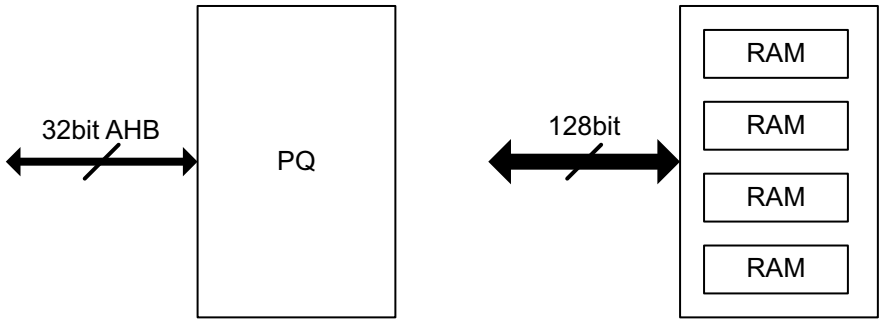


Fig 192. PowerQuad using RAM Bank 4 (16kB) as the128bit Wide RAM scratch pad

49.5.1 PowerQuad coprocessor operation

The ARMv8-M architecture (ARM Cortex-M33) introduces a coprocessor interface allowing PowerQuad access via MCR (Move from Coprocessor to Register) and MRC (Move from Register to Coprocessor) opcodes. Using this, up to two registers can be transferred between CM33 core and the PowerQuad.

On submitting the data and/or opcodes to a coprocessor, the CM33 can continue executing other tasks while the coprocessor computes in parallel.

In the PowerQuad, this interface is used for short (several cycles) operations for which the CM33 provides data from one of its registers and the result is stored back in the CM33 register.

When the data is provided to the PowerQuad in fixed point or floating point format, it is internally handled as floating point (conversion from fixed to floating point is done automatically if needed).

Result of PowerQuad operations can be fixed or floating point (conversion from floating to fixed point, if needed).

The functions handled in this way are:

- sin(x), cos(x), ln(x), e^(x), e^(-x),1/(x),1/sqrt(x), sqrt(x)
- biquad(x)

49.5.2 PowerQuad AHB operation

Large data operations are handled in a memory mapped fashion. The CM33 writes to the PowerQuad's Control registers to select the function, provide 4 memory regions with source/s destination addresses, and temporary scratch address for intermediate results (where applicable such as FFT, Matrix Inversion). These regions are: Input A, Input B, Output, and Temp

Table 1051.Register overview (base address 0x400A 6000)

Address	Name	Description	Access	Comments
0x000	OUTBASE	Base address register for output region	RW	Input A, B and Output region settings
0x004	OUTFORMAT	Data format for output region	RW	
0x008	TMPBASE	Base address register for temp region	RW	
0x00C	TMPFORMAT	Data format for region Temp	RW	
0x010	INABASE	Base address register for input A region	RW	
0x014	INAFORMAT	Data format for region input A	RW	
0x018	INBBASE	Base address register for input B region	RW	
0x01C	INBFORMAT	Data format for region input B	RW	
0x100	CONTROL	PowerQuad Control register	RW	Command Register
0x104	LENGTH	Length register	RW	0
0x108	CPPRE	Coprocessor Pre-scale register	RW	0
0x10C	MISC	Miscellaneous use register	RW	0
0x110	CURSORY	Cursory register	RW	Cordic Engine Input and Output registers
0x180	CORDIC_X	Cordic input X register	RW	
0x184	CORDIC_Y	Cordic input Y register	RW	
0x188	CORDIC_Z	Cordic input Z register	RW	
0x18C	ERRSTAT	Read/Write register where error statuses are captured (sticky)	RW1C	Interrupt and Status Flags
0x190	INTREN	Determines which conditions will assert the interrupt output.	RW	
0x194	EVENTEN	Determines which conditions will assert the Event Trigger output.	RW	
0x198	INTRSTAT	INTERRUPT STATUS register	RW1C	
0x200 - 0x23C	GPREGS[16]	General purpose register bank (16 x 32 bits)	RW	Computing Registers
0x240 - 0x25C	COMPREGS[8]	Compute register bank (8 x 32 bits)	RW	

Completion can be polled in the Control register INST_BUSY bit (cleared to '0'), or an interrupt/event from the PowerQuad can be generated.

The functions handled in this way are:

- Matrix (add, sub, scale, invert, transpose, multiply, dot, product)
- FFT/IFFT/DCT/IDCT (note that DCT is partial in the PowerQuad and requires some adjustments)
- FIR/ Convolution/ Correlation.

49.5.2.1 Simplified architecture

The block diagram here is a simplified representation of the PowerQuad's functional.

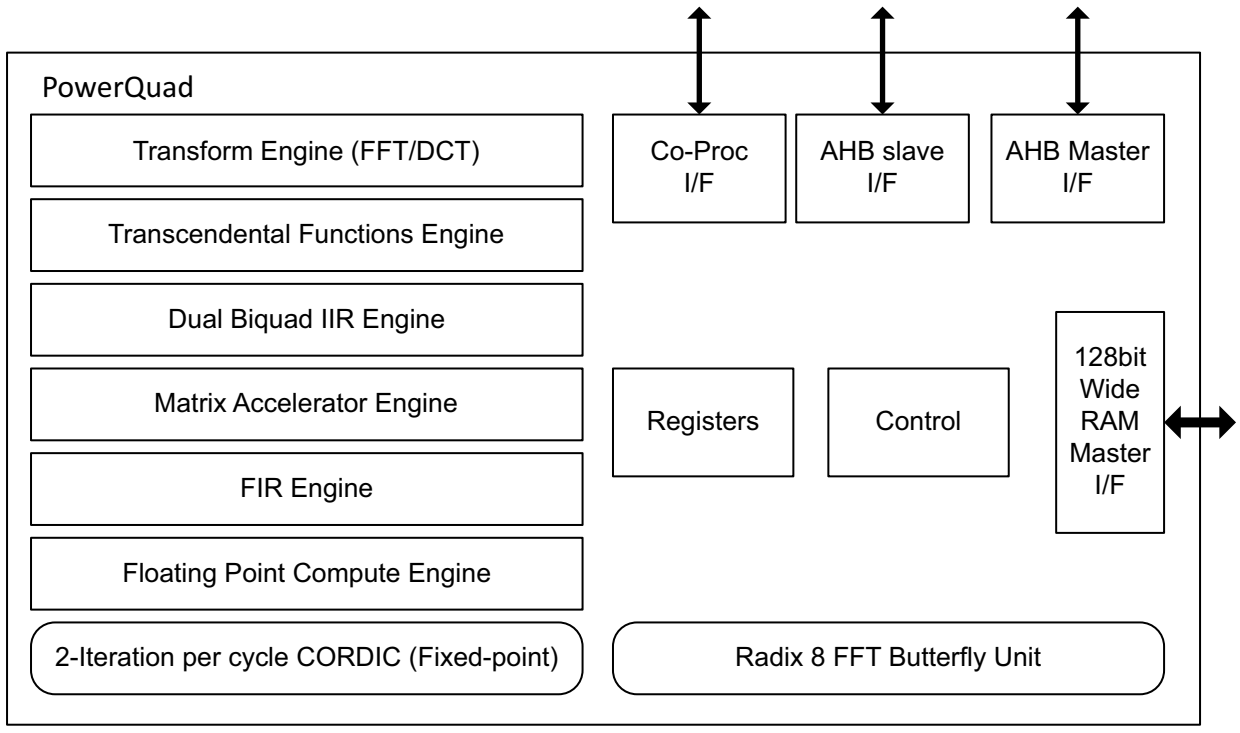


Fig 193. PowerQuad architecture

49.5.3 PowerQuad API functions

Supporting PowerQuad functions with driver APIs relieve the user from writing code to implement these functions.

NXP provides a comprehensive list of PowerQuad driver functions, which can be downloaded from SDK (search for “powerquad drivers” at www.nxp.com).

Also search for PowerQuad application notes where example code for various subsystems show how PowerQuad driver functions do the work.

PowerQuad operations are grouped into four families: Math, Filtering, Matrix, and Transform. The operations implemented in each family are listed in Table 1052.

Notes on the PowerQuad driver function table:

In the driver function table, the following terms are used:

- CP = coprocessor
- Xform = Transform
- Tran = Transcendental
- Trig = Trigonometry
- IIR = Infinite Impulse Response
- FIR = Finite Impulse Response
- FP = 32-bit single-precision floating point
- N.A.=Not Applicable
- Fix-16 = 16-bit fixed point
- Fix-32 = 32-bit Fixed Point
- Q1.31 = signed 32-bit fixed point with 31 bits to the right of the binary point

Table 1052.Summary of PowerQuad driver functions

Operation	Driver function	Access type	Input/Output data formats	InputA region usage	InputB region usage	Output region usage	Temp region usage	Fixed point input/ output scalers	Engine	Uses GPREGs/ COMPREGs
1/x	PQ_InvFixed PQ_InvF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
sqrt(x)	PQ_SqrtFixed PQ_SqrtF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
1/sqrt(x)	PQ_InvSqrtFixed PQ_InvSqrtF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
ln(x)	PQ_LnFixed PQ_LnF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
e^(x)	PQ_EtoxFixed PQ_EtoxF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
e^(-x)	PQ_EtonxFixed PQ_EtonxF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
x1 / x2	PQ_DivF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
sin(x)	PQ_SinFixed PQ_SinF32	CP	FP, Q1.31 (in radians, normalized)	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Trig	No
cos(x)	PQ_CosFixed PQ_CosF32	CP	FP, Q1.31 (in radians, normalized)	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Trig	No
IIR Filter (x)	PQ_*Biquad*	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	N.A.	cppre_in cppre_out	Biquad	Yes
arctan(x)	PQ_ArctanFixed	AHB	Fix-16, Fix-32	CORDIC_X	CORDIC_Y	CORDIC_Z	N.A.	N.A.	CORDIC	No
arctanh(x)	PQ_ArctanhFixed	AHB	Fix-16, Fix-32	CORDIC_X	CORDIC_Y	CORDIC_Z	N.A.	N.A.	CORDIC	No
FIR Filter	PQ_FIR	AHB	FP, Fix-16, Fix-32	Input data	Filter coefficients	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	FIR	No
FIR Filter Incremental	PQ_FIR	AHB	FP, Fix-16, Fix-32	Input data	Filter coefficients	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	FIR	No
Convolution	PQ_FIR	AHB	FP, Fix-16, Fix-32	Input data	Filter coefficients	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	FIR	No
Correlation	PQ_FIR	AHB	FP, Fix-16, Fix-32	Input data	Filter coefficients	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	FIR	No
Matrix Addition	PQ_MatrixAddition	AHB	FP, Fix-16, Fix-32	Matrix M1	Matrix M2	Result Matrix	N.A.	Ina_scaler Inb_scaler Out_scaler	Matrix	No

Table 1052.Summary of PowerQuad driver functions *…continued*

Operation	Driver function	Access type	Input/Output data formats	InputA region usage	InputB region usage	Output region usage	Temp region usage	Fixed point input/ output scalers	Engine	Uses GPREGs/ COMPREGs
Matrix Subtraction	PQ_MatrixSubtraction	AHB	FP, Fix-16, Fix-32	Matrix M1	Matrix M2	Result Matrix	N.A.	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Hadamard Product	PQ_MatrixMultiplication	AHB	FP, Fix-16, Fix-32	Matrix M1	Matrix M2	Result Matrix	N.A.	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Product	PQ_MatrixProduct	AHB	FP, Fix-16, Fix-32	Matrix M1	Matrix M2	Result Matrix	N.A.	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Invert	PQ_MatrixInversion	AHB	FP, Fix-16, Fix-32	Matrix M1	N.A.	Result Matrix	Uses max. 1024 words	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Transpose	PQ_MatrixTranspose	AHB	FP, Fix-16, Fix-32	Matrix M1	N.A.	Result Matrix	N.A.	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Scale	PQ_MatrixScale	AHB	FP, Fix-16, Fix-32	Matrix M1	N.A. (Scale factor in MISC register)	Result matrix	N.A.	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Vector Dot Product	PQ_VectorDotProduct	AHB	FP, Fix-16, Fix-32	Vector A	Vector B	Scaler result	N.A.	Ina_scaler Inb_scaler Out_scaler	Matrix	No
FFT of complex-valued input sequence	PQ_TransformCFFT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
FFT of real-valued input sequence	PQ_TransformRFFT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
Inverse FFT	PQ_TransformIFFT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
DCT of complex-valued input sequence	PQ_TransformCDCT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
DCT of real-valued input sequence	PQ_TransformRDCT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
Inverse DCT	PQ_TransformIDCT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	N.A.	Ina_scaler Inb_scaler Out_scaler	Xform	Yes

49.5.3.1 Example code for math function

Reciprocal (1/x)

```
/* Copyright 2018 NXP. NXP Confidential. This software is owned or controlled by NXP
     and may only be

* used strictly in accordance with the applicable license terms found at

* https://www.nxp.com/LA_OPT_NXP_SW. The "production use license" in Section 2.3 in
     the NXP SOFTWARE

* LICENSE AGREEMENT is expressly granted for this software.

*//*! Prototype
 * @brief Processing function for the fixed reciprocal.
 *
 * @param val value to be calculated
 * @return returns inv(val).
 */
static inline q31_t PQ_InvFixed(q31_t val)
/*! Prototype
 * @brief Processing function for the floating-point reciprocal.
 *
 * @param  *pSrc      points to the block of input data
 * @param  *pDst      points to the block of output data
 */
static inline void PQ_InvF32(float32_t *pSrc, float32_t *pDst)
```

Code for the invoking the API:

```
/* Copyright 2018 NXP. NXP Confidential. This software is owned or controlled by NXP
     and may only be

* used strictly in accordance with the applicable license terms found at

* https://www.nxp.com/LA_OPT_NXP_SW. The "production use license" in Section 2.3 in
     the NXP SOFTWARE

* LICENSE AGREEMENT is expressly granted for this software.

*//* Q15 Reciprocal */
    q15_t inputValue = 3;
    q15_t invResult = 5;

    /* Reciprocal */
    invResult = PQ_InvFixed(inputValue);
/* Q31 Reciprocal */
    q31_t inputValue = 3;
    q31_t invResult = 5;
    q31_t invRef = 0;
/* Reciprocal */
    invResult = PQ_InvFixed(inputValue);
/* Float Reciprocal */
    float input = 5.0;
```

```
    float Result;
PQ_InvF32(&input, &Result);
```

49.5.3.2 Example code for filtering functions

Table 1053.Second order IIR filter (single biquad section, direct-form II implementation

Supported formats	Cycle count (includes format convert + compute)	Accuracy (worst SNR) 10*log(((res-ref)/ref)^2)
Int32, Int16, FP	5 (1 cycle conversion + 4 cycles compute)	

The signal-flow graph of the direct-form II implementation of a 2nd order recursive digital filter is shown below.

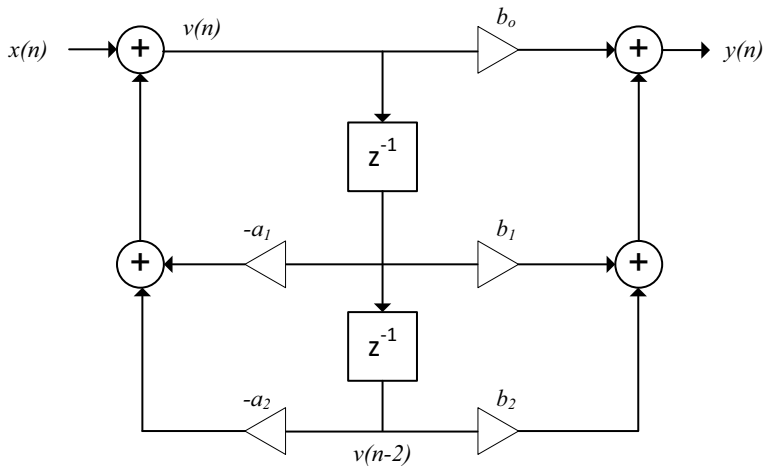


Fig 194. PowerQuad second order IIR filter

This filter implements the following difference equation:

v(n) = x(n) - a1v(n-1) - a2v(n-2)

y(n) = b0v(n) + b1v(n-1) + b2v(n-2)

Code for the invoking the API:

```
/* Copyright 2018 NXP. NXP Confidential. This software is owned or controlled by NXP
     and may only be

* used strictly in accordance with the applicable license terms found at

* https://www.nxp.com/LA_OPT_NXP_SW. The "production use license" in Section 2.3 in
     the NXP SOFTWARE

* LICENSE AGREEMENT is expressly granted for this software.

*/
/* Q31 biquad cascade IIR */
static void PQ_BiquadCascadedf2Q15Test(void)
{
    uint8_t stage = 3;
```

```
    /* 2 words current states:0, 0,
       2 words a cofficients:0x3e80b780(0.2514), 0x3f00b780(0.5028),
       3 words b
      cofficients:0x3e80b780(0.2514),0xbe2f4f0e(-0.1712),0x3e350b0f(0.1768),
       1 word previous addition:0.*/
    q31_t state[24] = {0, 0, 0x3e80b780, 0x3f00b780, 0x3e80b780, 0xbe2f4f0e,
     0x3e350b0f, 0,
                       0, 0, 0x3e80b780, 0x3f00b780, 0x3e80b780, 0xbe2f4f0e, 0x3e350b0f,
     0,
                       0, 0, 0x3e80b780, 0x3f00b780, 0x3e80b780, 0xbe2f4f0e, 0x3e350b0f,
     0};
    q15_t dataForBiquad[16] = {1024, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
    q15_t biquadResult[16] = {0};
    q15_t biquadRef[16] = {16, -46, 64, -35, -20, 45, -15, -26, 25, 6, -21, 5, 13, -8,
     -5, 7};

    pq_biquad_cascade_df2_instance instance;

    PQ_BiquadCascadeDf2Init(&instance, stage, (pq_biquad_state_t *)state);

#if 1
    PQ_BiquadCascadeDf2Q15(&instance, dataForBiquad, biquadResult, 15);
#else
    PQ_BiquadCascadeDf2Q15(&instance, dataForBiquad, biquadResult, 8);
    PQ_BiquadCascadeDf2Q15(&instance, &dataForBiquad[8], &biquadResult[8], 7);
#endif

    for (uint32_t i = 0; i < 15; i++)
    {
        TEST_ASSERT_TRUE(biquadRef[i] == biquadResult[i]);
    }
}
```

49.5.3.3 Example code for matrix functions

The Matrix accelerator engine supports the eight operations listed below, given with their respective maximum supported dimensions.

For Matrix of different sizes (larger), user will have to break up the Matrix into smaller block sizes and process block by block.

Matrix data are expected to be stored in memory row-by-row, arranged like standard C/C++ arrays.

So, if two 2x2 integer matrices A and B are

A = [1 2]

[3 4]

B = [5 6]

[7 8]

then the input data is expected to be stored in memory arrays as follows:

int data_A[4] = {1, 2, 3, 4};

int data_B[4] = {5, 6, 7, 8};

Matrix Addition, Matrix Subtraction

The addition (or subtraction) of two R-rows x C-columns matrices M1 and M2 is an R-row x C-column matrix M3 whose elements are the sums (or differences) of the corresponding input elements.

M1 rows must equal M2 rows, M1 cols must equal M2 cols.

Max. Rows = 16, max. cols = 16

Input data types: Float, Int32, Int16

LENGTH register configuration: LENGTH[23:16] = M2 cols (redundant in this case)

LENGTH[15:8] = M1 cols

LENGTH[7:0] = M1 rows

For instance:

M1				M2				M3		
1.1	2.2	3.3	+	.1	.2	.4	=	1.1 + .1	2.2 + .2	3.3 + .4
3.4	4.4	5.5		.3	.5	.6		3.4 + .3	4.4 + .5	5.5 + .6

Examples

```
/* Copyright 2018 NXP. NXP Confidential. This software is owned or controlled by NXP
     and may only be

* used strictly in accordance with the applicable license terms found at

* https://www.nxp.com/LA_OPT_NXP_SW. The "production use license" in Section 2.3 in
     the NXP SOFTWARE

* LICENSE AGREEMENT is expressly granted for this software.

*/
/* Q31 Matrix Addition */
static void PQ_MatrixAdditionQ31Test(void)
{
    int32_t length = (2 << 16) | (2 << 8) | (2 << 0);
    q31_t A31[4] = {1, 2, 3, 4};
    q31_t B31[4] = {1, 2, 3, 4};
    q31_t addResult31[4] = {0};
    q31_t addRef31[4] = {2, 4, 6, 8};

    PQ_SetFormat(POWERQUAD_NS, kPQ_CP_MTX, kPQ_32Bit);
```

```
    /* Matrix Addition */
    PQ_MatrixAddition(POWERQUAD_NS, length, (void *)A31, (void *)B31, (void
     *)addResult31);
    PQ_WaitDone(POWERQUAD_NS);

    for (uint32_t i = 0; i < DATA_SIZE; i++)
    {
        TEST_ASSERT_TRUE(addRef31[i] == addResult31[i]);
    }
}

}

/* F32 Matrix Substraction */
static void PQ_MatrixSubstractionF32Test(void)
{
    int32_t length = (2 << 16) | (2 << 8) | (2 << 0);
    float32_t M1[4] = {11.0, 22.0, 33.0, 44.0};
    float32_t M2[4] = {1.0, 2.0, 3.0, 4.0};
    float32_t subResult[4] = {0};
    float32_t subRef[4] = {10, 20, 30, 40};

    PQ_SetFormat(POWERQUAD_NS, kPQ_CP_MTX, kPQ_Float);

    /* Matrix Subtraction */
    PQ_MatrixSubtraction(POWERQUAD, length, (void *)M1, (void *)M2, (void
     *)subResult);
    PQ_WaitDone(POWERQUAD);

    for (uint32_t i = 0; i < DATA_SIZE; i++)
    {
        TEST_ASSERT_TRUE(subRef[i] == subResult[i]);
    }
}
```

49.5.4 Example code for transform functions

The PowerQuad provides discrete fourier transforms and discrete cosine transforms, implemented with a Radix-8 Butterfly structure fast fourier transform engine using fixed-point arithmetic at a resolution of 24 bits.

49.5.5 The discrete Fourier transform

The discrete Fourier transform (DFT) turns a sequence of complex numbers into another sequence of complex numbers:

$\{x_n\} := x_0, x_1, ..., x_{N-1}$ becomes $\{X_k\} := X_0, X_1, ..., X_{N-1}$

The transform is given by:

$$X_k = \sum_{n=0}^{N-1} x_n * e^{-i2\pi kn / N}$$

$$= \sum_{n=0}^{N-1} x_n * [\cos(2\pi kn / N) - i * \sin(2\pi kn / N)]$$

The inverse transform is given by:

$$x_n = \frac{1}{N} \sum_{n=0}^{N-1} X_k * e^{-i2\pi nk / N}$$

The real-input DFT

If x0, ..., xN-1 are real numbers, as they often are in practical applications, then the DFT obeys the symmetry:

$X_{N-k} \equiv X_{-k} = X^*_k$, where X^* denotes complex conjugation

It follows that X_0 and $X_{N/2}$ are real-valued, and the remainder of the DFT is completely specified by just N/2-1 complex numbers.

49.5.6 The discrete cosine transform

The discrete Cosine transform is a real and invertible function in which the N real numbers x0, x1, ... xN-1 are transformed into the N real numbers X0, X1, ... XN-1 .

DCT-II and DCT-III are implemented by the PowerQuad for forward and inverse DCT, where DCT-II is given by the formula:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right] \qquad k = 0, \ldots, N-1$$

and DCT-III (or the iDCT), which is the inverse of DCT-II, is given by the formula:

$$X_k = \frac{1}{2}x_0 + \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}n\left(k + \frac{1}{2}\right)\right] \qquad k = 0, \ldots, N-1$$

DCT-II implies the boundary conditions: xn is even around n = -1/2 and even around n = N-1/2; Xk is even around k = 0 and odd around k = N.

DCT-III implies the boundary conditions: xn is even around n = 0 and odd around n = N; Xk is even around k= -1/2 and odd around k = N-1/2.

Because of the real-input DFT symmetry, along with the implicit periodicity of the DFT, and the fact that the DFT of a real-even sequence is real and even, one can design an efficient DCT algorithm by eliminating the redundant operations from a larger FFT of real, symmetric data.

49.5.7 PowerQuad FFT and DCT implementation details

The PowerQuad FFT engine always scales the input data by 1/N when computing the FFT (and by extension DCT). If an unscale result is necessary, the input data (in the INPUT A region) must first be multiplied by N.

Note that the FFT engine only looks at the bottom 27 bits of the input word, so any pre-scaling must not exceed this in order to avoid saturation.

The inverse FFT is also scaled by 1/N, but this is correct as per the iDFT formula, so no scaling treatment is needed.

The supported lengths for PowerQuad FFTs / DCTs are N = 16, 32, 64, 128, 256, and 512 points.

The purely real (prefixed by ‘r’), and the complex flavors of the functions (prefixed by 'c') expect the input data sequences to be arranged in memory as follows.

If the sequence x = x0, x1 ... xN-1 are real numbers, then the input array in memory must be organized as x[N] = {x0, x1, ... xN-1};

If the sequence x = x0, x1 ... xN-1 are complex numbers of the form (x0real + x0im), (x1real + x1im), ... (xN-1real + xN-1im), then the input array in memory must be organized as

x[N] = {x0real, x0im, x1real, x1im, ... xN-1real, xN-1im};

The output sequence will always be stored in memory organized as an array of complex numbers where the imaginary parts will be zero for real-valued output data.

The PowerQuad does not perform the whole DCT or IDCT. It performs only the odd/even mirroring (or data folding) and FFT or IFFT. The final step, multiplication by the rotating phasor 2*e(-j*pi*k/(2*N)) must be performed by the CM33 to complete the transform.

For the forward DCT, the PowerQuad first performs odd/even mirroring on the input sequence. For example, [a b c d e f] becomes [a c e f d b]. Then, the FFT is performed, resulting in a complex-valued output sequence [A B C D C* B*]. To complete the DCT, the CM33 needs to multiply the output vector against the phasor rotating at 2N per circle.

For the inverse DCT, the CM33 needs to first multiply the input vector by the rotating phasor. The PowerQuad iDCT machine will then perform an inverse FFT, followed by reversal of the odd/even mirroring done by the forward transform.

49.5.8 Structure of the FFT engine

The Radix-8 butterfly structure of the engine is shown below. This implementation reduces memory accesses, and makes full use of the four multipliers available in the PowerQuad.

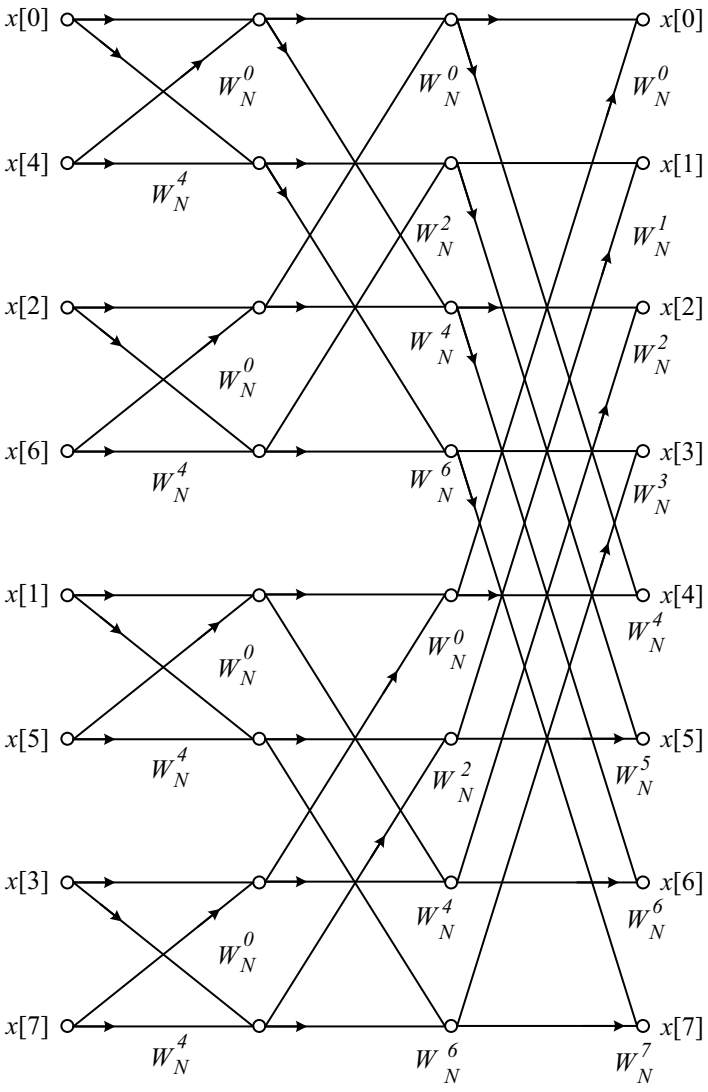


Fig 195. Radix-8 butterfly structure

49.5.9 Transform functions available in the PowerQuad

FFT

- PQ_TransformCFFT (complex-valued input sequence)
- PQ_TransformRFFT (real-valued input sequence)
- PQ_TransformIFFT (complex-valued input sequence, inverse FFT)

DCT

- PQ_TransformCDCT (complex-valued input sequence; engine ignores the imaginary part)
- PQ_TransformRDCT (real-valued input sequence)
- PQ_TransformIDCT (real-valued input sequence, inverse DCT)

49.5.10 Example code for transform function

```
/* Copyright 2018 NXP. NXP Confidential. This software is owned or controlled by NXP
     and may only be used strictly in accordance with the applicable license terms
     found at https://www.nxp.com/LA_OPT_NXP_SW. The "production use license" in
     Section 2.3 in the NXP SOFTWARE

* LICENSE AGREEMENT is expressly granted for this software.

*/

/* Q31 cfft */
static void PQ_CFFTQ31Test(void)
{
    int N = 32;
    q31_t inputData[64];
    q31_t cfftResult[64];
   q31_t cfftRef[64] = {100,  0,   76,   -50, 29,  -62, -1, -34, 10,  -3,  42, -8,  51,
     -47,  12,   -84,
                        -50,  -70, -82,  -4,  -46, 67,  40, 82,  109, 17,  98, -86, 5,
     -147, -110, -113,
                          -160, 0,   -110, 112, 5,   146, 98, 86,  109, -17, 40, -83,
     -46, -68,  -82,  4,
                        -50,  70,  12,   82,  51,  46,  42, 6,   10,  2,   -1, 33,  29,
     61,   77,   49};

    for (int i = 0; i < 64; i++)
    {
        inputData[i] = 0;
    }

    inputData[0] = 10;
    inputData[2] = 30;
    inputData[4] = 10;
    inputData[6] = 30;
    inputData[8] = -50;
    inputData[10] = 70;

    pq_config_t pq_cfg;

    pq_cfg.inputAFormat = kPQ_32Bit;
    pq_cfg.inputAPrescale =
        (N == 16) ? 4 : (N == 32) ? 5 : (N == 32) ? 5 : (N == 64) ? 6 : (N == 128) ? 7
     : (N == 256) ? 8 : 9;
    pq_cfg.inputBFormat = kPQ_32Bit;
    pq_cfg.inputBPrescale = 0;
    pq_cfg.tmpFormat = kPQ_32Bit;
    pq_cfg.tmpPrescale = 0;
    pq_cfg.outputFormat = kPQ_32Bit;
    pq_cfg.outputPrescale = 0;
    pq_cfg.tmpBase = (uint32_t *)0xe0000000;
```

```
    pq_cfg.machineFormat = kPQ_32Bit;
    PQ_SetConfig(POWERQUAD_NS, &pq_cfg);

    PQ_TransformCFFT(POWERQUAD_NS, N, inputData, cfftResult);
    PQ_WaitDone(POWERQUAD_NS);

    for (uint32_t i = 0; i < 64; i++)
    {
        TEST_ASSERT_TRUE(cfftRef[i] == cfftResult[i]);
    }
}


/* Q31 rfft */
static void PQ_RFFTQ31Test(void)
{
    int N = 32;
    q31_t inputData[32];
    q31_t rfftResult[64];
   q31_t rfftRef[64] = {100,  0,   76,   -50, 29,  -62, -1, -34, 10,  -3,  42, -8,  51,
     -47,  12,   -84,
                    -50,  -70, -82,  -4,  -46, 67,  40, 82,  109, 17,  98, -86, 5,
     -147, -110, -113,
                      -160, 0,   -110, 112, 5,   146, 98, 86,  109, -17, 40, -83,
     -46, -68,  -82,  4,
                    -50,  70,  12,   82,  51,  46,  42, 6,   10,  2,   -1, 33,  29,
     61,   77,   49};

    for (int i = 0; i < 32; i++)
    {
        inputData[i] = 0;
    }

    inputData[0] = 10;
    inputData[1] = 30;
    inputData[2] = 10;
    inputData[3] = 30;
    inputData[4] = -50;
    inputData[5] = 70;

    pq_config_t pq_cfg;

    pq_cfg.inputAFormat = kPQ_32Bit;
    pq_cfg.inputAPrescale =
        (N == 16) ? 4 : (N == 32) ? 5 : (N == 32) ? 5 : (N == 64) ? 6 : (N == 128) ? 7
     : (N == 256) ? 8 : 9;
    pq_cfg.inputBFormat = kPQ_32Bit;
    pq_cfg.inputBPrescale = 0;
    pq_cfg.tmpFormat = kPQ_32Bit;
    pq_cfg.tmpPrescale = 0;
    pq_cfg.outputFormat = kPQ_32Bit;
```

```
    pq_cfg.outputPrescale = 2;
    pq_cfg.tmpBase = (uint32_t *)0xe0000000;
    pq_cfg.machineFormat = kPQ_32Bit;
    PQ_SetConfig(POWERQUAD_NS, &pq_cfg);

    PQ_TransformRFFT(POWERQUAD_NS, N, inputData, rfftResult);
    PQ_WaitDone(POWERQUAD_NS);

    for (uint32_t i = 0; i < 64; i++)
    {
        TEST_ASSERT_TRUE(rfftRef[i] == rfftResult[i]);
    }
}


/* Q15 ifft */
static void PQ_IFFTQ15Test(void)
{
    int N = 32;
    q15_t ifftResult[64];
    q15_t inputData[64] = {100,  0,   76,   -50, 29,  -62, -1, -34, 10,  -3,  42, -8,
     51,  -47,  12,   -84,
                           -50,  -70, -82,  -4,  -46, 67,  40, 82,  109, 17,  98, -86,
     5,   -147, -110, -113,
                           -160, 0,   -110, 113, 5,   147, 98, 86,  109, -17, 40, -83,
     -46, -67,  -82,  4,
                           -50,  70,  12,   84,  51,  47,  42, 8,   10,  3,   -1, 34,
     29,  62,   76,   50};
    q15_t ifftRef[64] = {9, 0,  29, 0,  9, 0,  29, 0,  -51, -1, 69, -1, -1, 0,  -1, 0,
     0,  -1, 0,  -1, 0, -1,
                         0, -1, 0,  -1, 0, -1, 0,  -1, 0,   -1, 0,  -1, 0,  -1, 0,  -1,
     0,  -1, 0,  -1, 0, -1,
                         0, -1, 0,  -1, 0, -1, 0,  -1, 0,   -1, 0,  0,  0,  -1, 0,  0,
     -1, 0,  -1, 0};

    pq_config_t pq_cfg;

    pq_cfg.inputAFormat = kPQ_16Bit;
    pq_cfg.inputAPrescale = 0;
    pq_cfg.inputBFormat = kPQ_16Bit;
    pq_cfg.inputBPrescale = 0;
    pq_cfg.tmpFormat = kPQ_16Bit;
    pq_cfg.tmpPrescale = 0;
    pq_cfg.outputFormat = kPQ_16Bit;
    pq_cfg.outputPrescale = 0;
    pq_cfg.tmpBase = (uint32_t *)0xe0000000;
    pq_cfg.machineFormat = kPQ_32Bit;
    PQ_SetConfig(POWERQUAD_NS, &pq_cfg);

    PQ_TransformIFFT(POWERQUAD_NS, N, inputData, ifftResult);
    PQ_WaitDone(POWERQUAD_NS);
```

```
    for (uint32_t i = 0; i < 64; i++)
    {
        TEST_ASSERT_TRUE(ifftRef[i] == ifftResult[i]);
    }
}


/* Q15 CDCT */
static void PQ_CDCTQ15Test(void)
{
    int N = 32;
    int64_t acc0;
    int64_t acc1;
    int64_t acc2;
    const int32_t *twiddle_table = dct32_twiddle;
    q15_t inputData[64] = {4, 0, 3, 0, 5, 0, 10, 0, 0, 0, 0, 0, 0};
    q15_t dctResult[64];
    /* Round down and approaching 0 compared with matlab*/
    q15_t dctRef[64] = {3,  0, 5,  0, 4,  0, 3, 0, 2,  0, 1,  0, 0,  0, 0,  0, -1, 0,
     -2, 0, -2, 0,
                        -2, 0, -2, 0, -1, 0, 0, 0, 0,  0, 1,  0, 1,  0, 2,  0, 2,  0,
     2,  0, 1,  0,
                        1,  0, 0,  0, 0,  0, 0, 0, -1, 0, -1, 0, -1, 0, -1, 0, -1, 0,
     0,  0};

    pq_config_t pq_cfg;

    pq_cfg.inputAFormat = kPQ_16Bit;
    pq_cfg.inputAPrescale =
        (N == 16) ? 4 : (N == 32) ? 5 : (N == 32) ? 5 : (N == 64) ? 6 : (N == 128) ? 7
     : (N == 256) ? 8 : 9;
    pq_cfg.inputBFormat = kPQ_16Bit;
    pq_cfg.inputBPrescale = 0;
    pq_cfg.tmpFormat = kPQ_16Bit;
    pq_cfg.tmpPrescale = 0;
    pq_cfg.outputFormat = kPQ_16Bit;
    pq_cfg.outputPrescale = 0;
    pq_cfg.tmpBase = (uint32_t *)0xe0000000;
    pq_cfg.machineFormat = kPQ_32Bit;
    PQ_SetConfig(POWERQUAD_NS, &pq_cfg);

    PQ_TransformCDCT(POWERQUAD_NS, N, inputData, dctResult);
    PQ_WaitDone(POWERQUAD_NS);

    for (int i = 0; i < N; i++)
    {
       acc0 = (int64_t)dctResult[i * 2] * twiddle_table[i * 2];            /* real *
     real */
       acc1 = (int64_t)dctResult[(i * 2) + 1] * twiddle_table[(i * 2) + 1]; /*
     imaginary * imaginary */
```

```
        acc2 = acc0 - acc1;
        dctResult[i * 2] = (uint32_t)(acc2 / (1024 * 1024 * 16));
        dctResult[(i * 2) + 1] = 0; /* zero out imaginary */
    }

    for (uint32_t i = 0; i < 64; i++)
    {
        TEST_ASSERT_TRUE(dctRef[i] == dctResult[i]);
    }
}



/* Q31 RDCT */
static void PQ_RDCTQ31Test(void)
{
    int N = 32;
    int64_t acc0;
    int64_t acc1;
    int64_t acc2;
    const int32_t *twiddle_table = dct32_twiddle;
    q31_t inputData[32] = {4, 3, 5, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                           0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
    q31_t dctResult[64];
    /* Round down and approaching 0 compared with matlab*/
    q31_t dctRef[64] = {3,  0, 5,  0, 4,  0, 3, 0, 2,  0, 1,  0, 0,  0, 0,  0, -1, 0,
     -2, 0, -2, 0,
                        -2, 0, -2, 0, -1, 0, 0, 0, 0,  0, 1,  0, 1,  0, 2,  0, 2,  0,
     2,  0, 1,  0,
                        1,  0, 0,  0, 0,  0, 0, 0, -1, 0, -1, 0, -1, 0, -1, 0, -1, 0,
     0,  0};

    pq_config_t pq_cfg;

    pq_cfg.inputAFormat = kPQ_32Bit;
    pq_cfg.inputAPrescale =
        (N == 16) ? 4 : (N == 32) ? 5 : (N == 32) ? 5 : (N == 64) ? 6 : (N == 128) ? 7
     : (N == 256) ? 8 : 9;
    pq_cfg.inputBFormat = kPQ_32Bit;
    pq_cfg.inputBPrescale = 0;
    pq_cfg.tmpFormat = kPQ_32Bit;
    pq_cfg.tmpPrescale = 0;
    pq_cfg.outputFormat = kPQ_32Bit;
    pq_cfg.outputPrescale = 0;
    pq_cfg.tmpBase = (uint32_t *)0xe0000000;
    pq_cfg.machineFormat = kPQ_32Bit;
    PQ_SetConfig(POWERQUAD_NS, &pq_cfg);

    PQ_TransformRDCT(POWERQUAD_NS, N, inputData, dctResult);
    PQ_WaitDone(POWERQUAD_NS);
```

```
    for (int i = 0; i < N; i++)
    {
       acc0 = (int64_t)dctResult[i * 2] * twiddle_table[i * 2];            /* real *
     real */
       acc1 = (int64_t)dctResult[(i * 2) + 1] * twiddle_table[(i * 2) + 1]; /*
     imaginary * imaginary */
       acc2 = acc0 - acc1;
       dctResult[i * 2] = (uint32_t)(acc2 / (1024 * 1024 * 16));
       dctResult[(i * 2) + 1] = 0; /* zero out imaginary */
    }

    for (uint32_t i = 0; i < 64; i++)
    {
        TEST_ASSERT_TRUE(dctRef[i] == dctResult[i]);
    }
}



/* Q15 IDCT */
static void PQ_IDCTQ15Test(void)
{
    int N = 32;
    int64_t acc0;
    int64_t acc1;
    int64_t acc2;
    int64_t tmp_re, tmp_im;
    const int32_t *twiddle_table = idct32_twiddle;
    q31_t tmp[64];
    q31_t inputData[64] = {3,  0, 5,  0, 4,  0, 3, 0, 2,  0, 1,  0, 0,  0, 0,  0, -1,
     0, -2, 0, -2, 0,
                           -2, 0, -2, 0, -1, 0, 0, 0, 0,  0, 1,  0, 1,  0, 2,  0, 2,
     0, 2,  0, 1,  0,
                           1,  0, 0,  0, 0,  0, 0, 0, -1, 0, -1, 0, -1, 0, -1, 0, -1,
     0, 0,  0};
    q31_t idctResult[64];
    q31_t idctRef[64] = {3,  1,  2,  0,  3,  0,  7,  0,  -1, 0,  0,  -1, -1, 0,  0,  0,
     -1, -1, 0, 0,  -1, -1,
                         -1, -1, -1, -1, 0,  -1, -1, 0,  -1, -1, 0,  -1, 0,  -1, 0,  -1,
     0,  -1, 0, -1, -1, -1,
                          0,  -1, 0,  -1, -1, -1, -1, -1, 0,  -1, -1, -1, 0,  -1, -1,
     -1, -1, -1, 0, -1};

    pq_config_t pq_cfg;

    pq_cfg.inputAFormat = kPQ_32Bit;
    pq_cfg.inputAPrescale = 0;
    pq_cfg.inputBFormat = kPQ_32Bit;
    pq_cfg.inputBPrescale = 0;
    pq_cfg.tmpFormat = kPQ_32Bit;
    pq_cfg.tmpPrescale = 0;
```

```
    pq_cfg.outputFormat = kPQ_32Bit;
    pq_cfg.outputPrescale = 0;
    pq_cfg.tmpBase = (uint32_t *)0xe0000000;
    pq_cfg.machineFormat = kPQ_32Bit;
    PQ_SetConfig(POWERQUAD_NS, &pq_cfg);

    for (int i = 0; i < N; i++)
    {
        if (i == 0)
        {
            tmp_re = inputData[0];
            tmp_im = 0;
        }
        else
        {
            tmp_re = inputData[i * 2];
            tmp_im = inputData[N * 2 - (i * 2)] * -1;
        }

        acc0 = tmp_re * twiddle_table[i * 2];       /* real * real */
        acc1 = tmp_im * twiddle_table[(i * 2) + 1]; /* imaginary * imaginary */
        acc2 = acc0 - acc1;
        tmp[i * 2] = (uint32_t)(acc2 / (1024 * 1024 * 16));

        acc0 = tmp_re * twiddle_table[(i * 2) + 1]; /* real * imaginary */
        acc1 = tmp_im * twiddle_table[(i * 2)];     /* imaginary * real */
        acc2 = acc0 + acc1;
        tmp[(i * 2) + 1] = (uint32_t)(acc2 / (1024 * 1024 * 16));
    }

    PQ_TransformIDCT(POWERQUAD_NS, N, tmp, idctResult);
    PQ_WaitDone(POWERQUAD_NS);

    for (uint32_t i = 0; i < 64; i++)
    {
        TEST_ASSERT_TRUE(idctRef[i] == idctResult[i]);
    }
}
```

49.5.11 Macros for the SDK examples

The SDK driver APIs need the following data structures defined

```
typedef enum
{
    kPQ_CP_PQ = 0,    /*!< Math engine.*/
    kPQ_CP_MTX = 1,   /*!< Matrix engine.*/
    kPQ_CP_FFT = 2,   /*!< FFT engine.*/
    kPQ_CP_FIR = 3,   /*!< FIR engine.*/
    kPQ_CP_CORDIC = 5 /*!< CORDIC engine.*/
} pq_computationengine_t;
```

```
/*! @brief powerquad data structure format type */
typedef enum
{
    kPQ_16Bit = 0, /*!< Int16 Fixed point.*/
    kPQ_32Bit = 1, /*!< Int32 Fixed point.*/
    kPQ_Float = 2  /*!< Float point.*/
} pq_format_t;

/*! @brief Coprocessor prescale */
typedef struct
{
    uint16_t inputPrescale;  /*!< Input prescale.*/
    uint16_t outputPrescale; /*!< Input prescale.*/
    uint16_t outputSaturate; /*!< Output saturate at n bits, for example 0x11 is 8 bit
     space,
                                                the value will be truncated at +127 or
     -128.*/
} pq_prescale_t;

/*! @brief powerquad data structure format */
typedef struct
{
    pq_format_t inputAFormat;  /*!< Inputa format.*/
    int32_t inputAPrescale;    /*!< Inputa prescale, for example 1.5 can be 1.5*2^n if
     you scale by 'shifting' ('scaling' by a factor of n).*/
     pq_format_t inputBFormat;  /*!< Inputb format.*/
    int32_t inputBPrescale;    /*!< Inputb inputb_prescale.*/
    pq_format_t outputFormat;  /*!< Out format.*/
    int32_t outputPrescale;    /*!< Out prescale.*/
    pq_format_t tmpFormat;     /*!< Temp format.*/
    int32_t tmpPrescale;       /*!< Temp prescale.*/
    pq_format_t machineFormat; /*!< Machine format.*/
    uint32_t *tmpBase;         /*!< Tmp base address.*/
} pq_config_t;

typedef struct _pq_biquad_state
{
    int32_t context[8]; /*!< Current states:2 words, a cofficients:2 words, b
     cofficients:3 words, previous addition:1 word.*/
} pq_biquad_state_t;

/*! @brief Instance structure for the direct form II Biquad cascade filter */
typedef struct
{
    uint8_t numStages;         /**< Number of 2nd order stages in the filter.*/
    pq_biquad_state_t *pState; /**< Points to the array of state coefficients.*/
} pq_biquad_cascade_df2_instance;

/*! @brief CORDIC iteration */
typedef enum
{
```

```
    kPQ_Iteration_8 = 0, /*!< Iterate 8 times.*/
    kPQ_Iteration_16,    /*!< Iterate 16 times.*/
    kPQ_Iteration_24     /*!< Iterate 24 times.*/
} pq_cordic_iter_t;

typedef struct
{
    float denominator;
    float numerator;
} float_divider_in_nums;

typedef union
{
    unsigned long long longval;
    float_divider_in_nums fval;
} fordiv_t;Get default configuration.

/*!
   * This function initializes the POWERQUAD configuration structure to a default
     value.
   * FORMAT register field definitions
   *   Bits[15:8] scaler (for scaled 'q31' formats)
   *   Bits[5:4] external format. 00b=q15, 01b=q31, 10b=float
   *   Bits[1:0] internal format. 00b=q15, 01b=q31, 10b=float
   *   POWERQUAD->INAFORMAT = (config->inputAPrescale << 8) | (config->inputAFormat <<
     4) | config->machineFormat
   * For all Powerquad operations internal format must be float (with the only
     exception being
   * the FFT related functions, ie FFT/IFFT/DCT/IDCT which must be set to q31).
   * The default values are:
   *   config->inputAFormat = kPQ_Float;
   *   config->inputAPrescale = 0;
   *   config->inputBFormat = kPQ_Float;
   *   config->inputBPrescale = 0;
   *   config->outputFormat = kPQ_Float;
   *   config->outputPrescale = 0;
   *   config->tmpFormat = kPQ_Float;
   *   config->tmpPrescale = 0;
   *   config->machineFormat = kPQ_Float;
   *
   * @param config Pointer to "pq_config_t" structure.
   */
void PQ_GetDefaultConfig(pq_config_t *config);

/*!
    * @brief Set configuration with format/prescale.
    *
    * @param base  POWERQUAD peripheral base address
    * @param config Pointer to "pq_config_t" structure.
    */
    void PQ_SetConfig(POWERQUAD_Type *base, const pq_config_t *config);
```

```
/*!
    * @brief set coprocessor scaler for coprocessor instructions, this function is
     used to
    * set output saturation and scaleing for input/output.
    *
    * @param base  POWERQUAD peripheral base address
    * @param prescale Pointer to "pq_prescale_t" structure.
    */
void PQ_SetCoprocessorScaler(POWERQUAD_Type *base, const pq_prescale_t *prescale);

    /*!
    * @brief Initializes the POWERQUAD module.
    *
    * @param base   POWERQUAD peripheral base address.
    */
void PQ_Init(POWERQUAD_Type *base);

/*!
    * @brief De-initializes the POWERQUAD module.
    *
    * @param base POWERQUAD peripheral base address.
    */
void PQ_Deinit(POWERQUAD_Type *base);

/*!
    * @brief Set format for non-coprocessor instructions.
    *
    * @param base  POWERQUAD peripheral base address
    * @param engine Computation engine
    * @param format Data format
*/
void PQ_SetFormat(POWERQUAD_Type *base, pq_computationengine_t engine, pq_format_t
     format);

/*!
    * @brief Wait for the completion.
    *
    * @param base  POWERQUAD peripheral base address
    */
void PQ_WaitDone(POWERQUAD_Type *base);

/*!
    * @brief Processing function for the floating-point natural log.
    *
    * @param  *pSrc      points to the block of input data
    * @param  *pDst      points to the block of output data
    */
```

50.1 How to read this chapter

The CASPER peripheral is available on all LPC55S6x/LPC55S2x devices. The Cryptographic Accelerator (CASPER) engine provides acceleration of asymmetric cryptographic algorithms.

When the Cryptographic Accelerator (CASPER) is used in conjunction with hardware blocks for hashing and symmetric cryptography, a potential performance boost can be achieved.

Supported crypto functions are implemented in the SDK (Software Development Kit) and the mbed TLS examples utilize the CASPER peripheral for computations. CASPER interface uses SRAMX_0 (0x1400 0000 to 0x1400 0FFF) and SRAMX_1 (0x1400 4000 to 0x1400 4FFF).

50.2 CASPER features

When relying on just the MCU without CASPER assistance, the ARM M33 must perform all of the computational processing which can potentially slow down some applications with intensive processing requirements. Under these conditions, CASPER improves performance and frees up the ARM M33 to perform other tasks.

CASPER provides acceleration of RSA, DH and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically, it enhances the performance of the following operations:

- RSA modular exponentiation
- ECC scalar multiplication, point on curve check
- ECDSA signature generation and verification

Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication are all performed in an accelerated manner.

The CASPER engine can compute assign functions that can be up to eight times faster than the ARM M33. See Table 1054 for a performance comparison for the various functions:

Table 1054.

Operation	Algorithm	Software only version CortexM33@100 MHz	With CASPER acceleration CortexM33@100 MHz	Performance improvement
Signing	ECDSA-secp256r1	258.5 ms	57.7 ms	4.48 times
Verification	ECDSA-secp256r1	500 ms	58.59 ms	8.53 times
Key exchange	ECDHE-secp256r1	469 ms	92.59 ms	5.07 times
Key exchange	ECDH-secp256r1	250 ms	46.88 ms	5.33 times

50.3 CASPER Operation

The CASPER module consist of:

- 4 x 32-bits (ABCD) Data Registers, feeding the two multipliers.
- A mask register used for creating an XOR mask for unmasking ABCD and masking output for side channel protection.
- The Multiplier has a special 1st *sum* mechanism to add the inner products back to complete a 64x64 multiply. It is much faster than a separated adder.
- 4 (RES[0-3]) Results registers which can be used with four adders, do Add-Mask and XOR operations.
- Special access to 2 RAMs (up to 4kB) in parallel.
  - The block can access these 2 RAM banks simultaneously, allowing for 2 (64 bits at a time) operations to have happen in parallel.
- Interleaving RAMs (i.e., one for the even words and one for the odd words) allows 64 bit word pairs to be accessed simultaneously. Note that, SYSCON.CASPER_CTRL is used to enable the interleaved addressing on RAMX0 to RAMX1, but it should be kept in mind when using these bits, that the RAMX interface for the CPU is also affected.
- A clock enable model is supported so that when the speed of the clock is faster than the pipe, the clock enables support for an MCP (multi-cyclic path) approach to completion of the operations.

The following block diagram shows a conceptual representation of how CASPER operations are performed through the data and result registers and how the mask is applied:

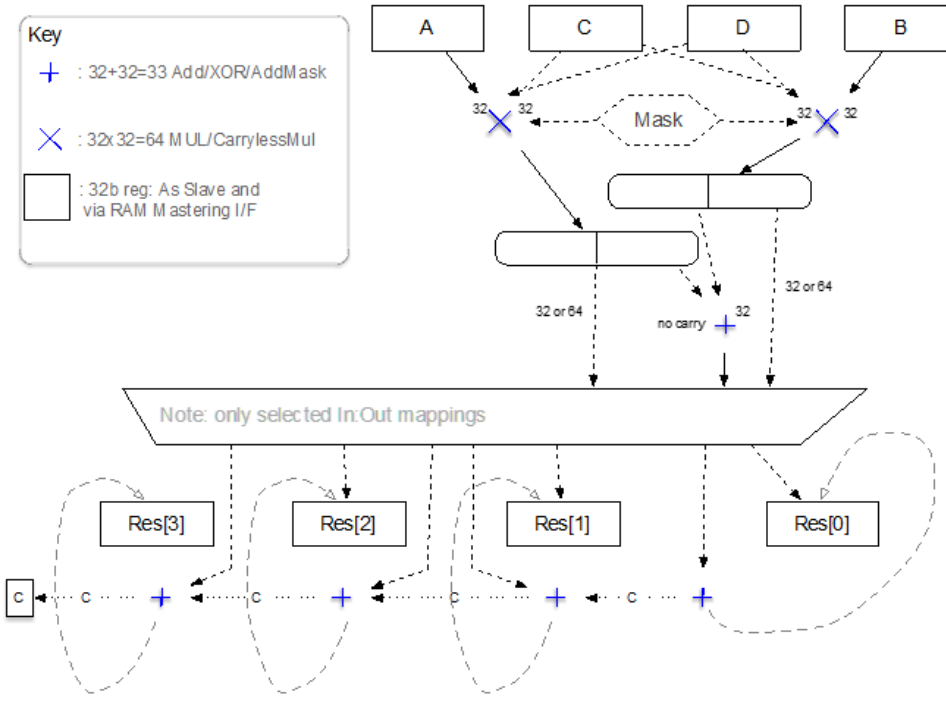


Fig 196. CASPER block diagram

CASPER uses SRAM Bank X (8kB at 0x0400 0000 to 0x0400 1FFF or 0x1400 0000 to 1FFF depending on the secured or Non-secured configuration) as an internal scratch pad as shown in the following example:

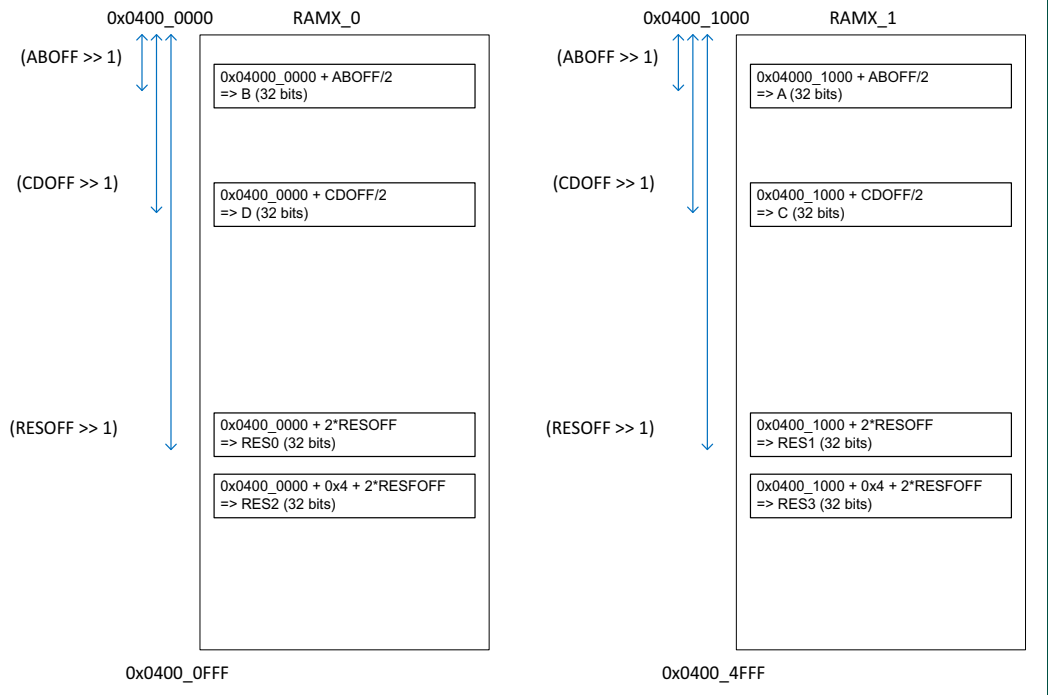


Fig 197. CASPER Memory map

This following example shows how CASPER registers are stored and used in memory:

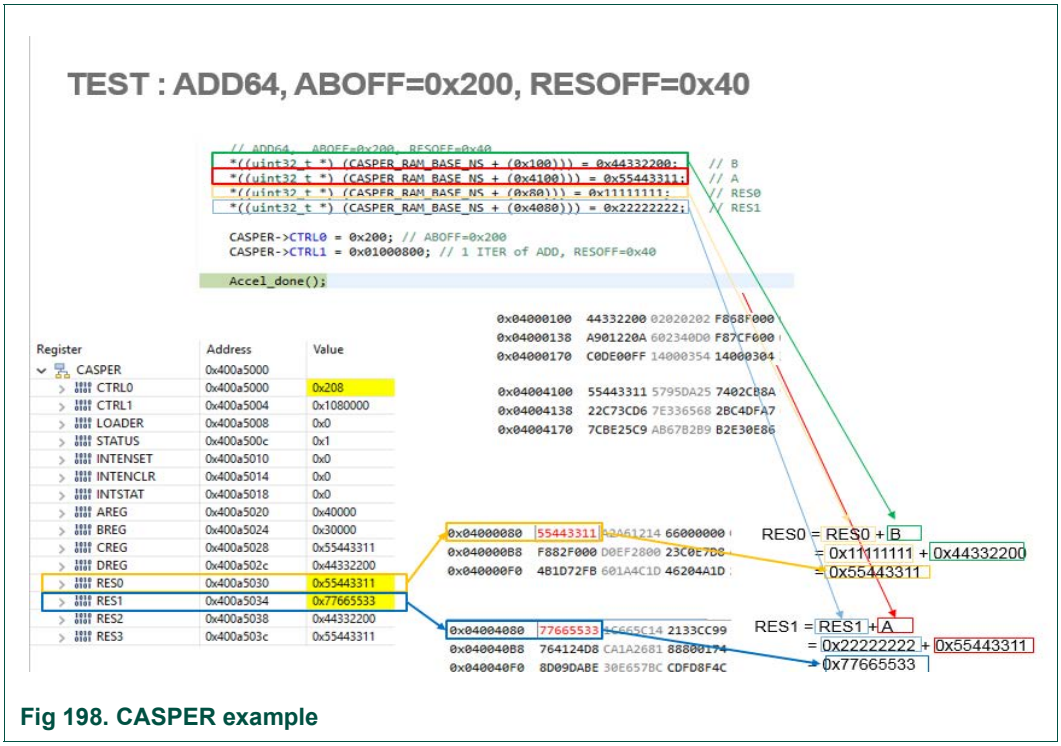


Fig 198. CASPER example

Based on this example:

- A & B are stored at (ABOFF >> 1):
  - B is stored at 0x0400_0000+(ABOFF >> 1)
  - A is stored at 0x0400_1000+(ABOFF >> 1)
- C & D are stored at (CDOFF >> 1):
  - D is stored at 0x0400_0000+(CDOFF >> 1)
  - C is stored at 0x0400_1000+(CDOFF >> 1)
- RESOFF is stored at (RESOFF >> 1):
  - RESO is stored at 0x0400_0000+(RESOFF >> 1)
  - RES1 is stored at 0x0400_1000+(RESOFF >> 1)
  - RES2 is stored at 0x0400_0000+(RESOFF >> 1)+0x4
  - RES3 is stored at 0x0400_1000+(RESOFF >> 1)+0x4

50.3.1 CASPER co-processor operation

The ARMv8-M architecture (ARM M33) provides a co-processor interface that allows access to CASPER via MCR (Move from Coprocessor to Register) and MRC (Move from Register to Coprocessor) opcodes. Through this interface, up to two registers can be transferred between the ARM M33 core and CASPER.

Upon submitting the data and/or opcodes to a co-processor, the ARM M33 can continue executing other tasks while the co-processor performs it computations in parallel.

The CRm register index is used as the word address, while CRn serves as the *bank*. For example, RES2 is offset 0x38 in memory, and CRm=(0x38>>2)=0xE, thus allowing access to registers between offset 0x00 and 0x3C with CRn=0x0. To access the higher registers, the CRn register can be set to 1, 2, or 3. For example, MASK can be accessed using CRn=1, and CRm=8. This can be computed as 0x60>>2=0x18, where the lower nibble goes into CRm, and the upper nibble into CRn.

The MCRR instruction allows access to pairs, including CTRL0/CTRL1, A/B, C/D, RES0/RES1, RES2/RES3 only. The lower index of the pair is used in CRm, with MCR.

50.3.2 CASPER AHB operation

A bus slave (AHB) and ARM M33 CP interface is provided so applications can access the control and data/results registers as needed. It can set up the operation, the iteration count, and the offset registers (offsets into the RAMs) and is optimized to allow two words (for example STRD or STM with memory, MCRR with CP) to perform the configuration and start in one write.

It may access the data and result registers when needed.

50.3.3 CASPER modes

Coarsely integrated operand scanning = CIOS

Most Significant Word = MSW

Least Significant Word = LSW

Table 1055. Casper AHB operations

Mode	Name	Description	Comments
0x01	MUL6464_NOSUM	Walking 1 or more of J loop, doing w[j]=ab*cd[j] base on 64x64=128.	Writes out results, but does not read in to add.
0x02	MUL6464_SUM	Walking 1 or more of J loop, doing c,w[j]=w[j]+ab*cd[j] base on 64x64=128.	Sums by reading result word (w[j]) and adding before writing back. This does not read the final 2 words before writing, since it is assumed they are farthest reached (w[i+j]).
0x03	MUL6464_FULLSUM	Walking 1 or more of J loop, doing c,w[j]=w[j]+ab*cd[j] base on 64x64=128, sum all of w.	Reads all of w, including the MSWs, it can have a carry in the carry bit Includes 1st loop half of CIOS multiply use (before reduce).

Table 1055. Casper AHB operations *…continued*

Mode	Name	Description	Comments
0x04	MUL6464_REDUCE	Walking 1 or more of J loop, doing c,w[j-1]=w[j]+m*cd[j] base on 64x64=128, but skip 1st write. Note that m is in ab and is w[0]*Np.	• Does not compute m into ab, need to this first. • The reduction pass of a CIOS double J loop. So, it writes to the preceding result double-word, except the 1st time, where it throws away the low-order 64 bits. • The full Montgomery use is FULLSUM 1st (1st J loop), then the processor computes the first product result of w[0]*Np64 (modulus N` as a 64 bit value). • This is similar to the MUL6464_FULLSUM operation except it skips the first write so it can write to the previous.
0x08	ADD64	ADD with ABOFF, and in/out RESOFF base on c,r=r+a+c.	Uses 64 bits at a time, producing 65 bit output. Final carry in the carry bit.
0x09	SUB64	SUBTRACT with ABOFF, and in/out RESOFF base on r=r-a.	Uses 64 bits at a time, and if a is larger, final borrow is implicit, but carry bit set if borrowed. Solved using r=r+(~a+1).
0x0C	RSUB64	SUBTRACT with ABOFF, and in/out RESOFF base on r=r-a.	Uses 64 bits at a time, and if a is larger, final borrow is implicit, but carry bit set if borrowed. Solved using r=r+(~a+1).
0x0A	DOUBLE64	ADD to self with RESOFF base on c,r=r+r+c.	Doubles a value, same as *2 or <<1 Uses 64 bits at a time, producing a 65 bit output, with final carry in the carry bit.
0x0B	XOR64	XOR with ABOFF, and in/out RESOFF base on r=r^a.	Uses 64 bits at a time, producing 64 bit output. No carry bit.
0x14	COPY	Copy from ABOFF to RESOFF using 64 bits at a time.	
0x16	FILL	Fill RESOFF with value in A and B, 64 bits at a time.	
0x17	ZERO	Fill RESOFF WITH 0s, 64 bits at a time.	

50.4 Register descriptions

Register descriptions for CASPER are provided in the following table.

Table 1056. Register overview: CASPER (base address 0x400A5000)

Name	Access	Offset	Description	Reset value	Section
CTRL0	R/W	0x0	Contains the offsets of AB and CD in the RAM.	undefined	50.4.1
CTRL1	R/W	0x4	Contains the opcode mode, iteration count, and result offset (in RAM) and also launches the accelerator. Note: with CP version: CTRL0 and CRTL1 can be written in one go with MCRR.	undefined	50.4.2
LOADER	R/W	0x8	Contains an optional loader to load into CTRL0/1 in steps to perform a set of operations.	undefined	50.4.3
STATUS	R/W	0xC	Indicates operational status and would contain the carry bit if used.	undefined	50.4.4
INTENSET	R/W	0x10	Sets interrupts.	undefined	50.4.5
INTENCLR	R/W	0x14	Clears interrupts.	undefined	50.4.6
INTSTAT	R/W	0x18	Interrupt status bits (mask of INTENSET and STATUS).	undefined	50.4.7
AREG	R/W	0x20	A register.	undefined	50.4.8
BREG	R/W	0x24	B register.	undefined	50.4.8
CREG	R/W	0x28	C register.	undefined	50.4.8
DREG	R/W	0x2C	D register.	undefined	50.4.8
RES0	R/W	0x30	Result register 0.	undefined	50.4.9
RES1	R/W	0x34	Result register 1.	undefined	50.4.9
RES2	R/W	0x38	Result register 2.	undefined	50.4.9
RES3	R/W	0x3C	Result register 3.	undefined	50.4.9
MASK	R/W	0x60	Optional mask register.	undefined	50.4.10
REMASK	R/W	0x64	Optional re-mask register.	undefined	50.4.11
LOCK	R/W	0x80	Security lock register.	undefined	50.4.12

50.4.1 Control 0 pin register

Contains the offsets of AB and CD in the RAM.

Table 1057. Contains the offsets of AB and CD in the RAM. (CTRL0, offset 0x0)

Bit	Symbol	Value	Description	Reset value
0	ABBPAIR		Which bank-pair the offset ABOFF is within. This must be 0 if only 2-up.	0x0
		0	Bank-pair 0 (1st)	
		1	Bank-pair 1 (2nd)	
1	-	-	Reserved.	undefined
2	ABOFF		Word or DWord Offset of AB values, with B at [2]=0 and A at [2]=1 as far as the code sees (normally will be an interleaved bank so only sequential to AHB). Word offset only allowed if 32 bit operation. Ideally not in the same RAM as the CD values if 4-up.	0x0
15:3				
16	CDBPAIR		Which bank-pair the offset CDOFF is within. This must be 0 if only 2-up.	
		0	Bank-pair 0 (1st)	
		1	Bank-pair 1 (2nd)	
17	-	-	Reserved	undefined
28:18	CDOFF		Word or DWord Offset of CD, with D at [2]=0 and C at [2]=1 as far as the code sees (normally will be an interleaved bank so only sequential to AHB). Word offset only allowed if 32 bit operation. Ideally not in the same RAM as the AB values.	
31:29	-	-	Reserved	undefined

50.4.2 Control 1 pin register

Contains the opcode mode, iteration count, and result offset (in RAM) and also launches the accelerator.

Note: with CP version: CTRL0 and CRTL1 can be written in one go with MCRR.

Table 1058. Contains the opcode mode, iteration count, and result offset (in RAM) and also launches the accelerator. (CTRL1, offset 0x4)

Bit	Symbol	Value	Description	Reset value
7:0	ITER		Iteration counter. Is number_cycles ñ 1. write 0 means Does one cycle ñ does not iterate.	0x0
15:8	MODE		Iteration counter. Is number_cycles ñ 1. write 0 means Does one cycle ñ does not iterate.	0x0
16	RESBPAIR		Which bank-pair the offset RESOFF is within. This must be 0 if only 2-up. Ideally this is not the same bank as ABBPAIR (when 4-up supported).	
		0	Bank-pair 0 (1st).	
		1	Bank-pair 1 (2nd).	

Table 1058. Contains the opcode mode, iteration count, and result offset (in RAM) and also launches the accelerator. (CTRL1, offset 0x4)

Bit	Symbol	Value	Description	Reset value
17	-	-	Reserved.	undefined
28:18	RESOFF		Word or DWord Offset of result. Word offset only allowed if 32 bit operation. Ideally not in the same RAM as the AB and CD values.	
31:29	-	-	Reserved	undefined

50.4.3 Loader register

Provides an optional loader to load into CTRL0/1 in order to perform a series of operations in succession.

Table 1059. Contains an optional loader to load into CTRL0/1 in steps to perform a set of operations. (LOADER, offset 0x8)

Bit	Symbol	Value	Description	Reset value
7:0	COUNT		Number of control pairs to load 0 relative (so 1 means load 1). write 1 means Does one open does not iterate, write N means N control pairs to load.	0x0
15:8			Reserved.	Undefined.
16	CTRLBPAIR		Which bank-pair the offset CTRLOFF is within. This must be 0 if only 2-up. Does not matter which bank is used as this is loaded when not performing an operation.	
		0	Bank-pair 0 (1st).	
		1	Bank-pair 1 (2nd).	
17				
28:18	CTRLOFF		DWord Offset of CTRL pair to load next.	0x0
31:29	-	-	Reserved	Undefined.

50.4.4 Status register

Indicates operational status and contains the carry bit if used.

Table 1060. Indicates operational status. (STATUS, offset 0xC)

Bit	Symbol	Value	Description	Reset value
0	DONE		Indicates if the accelerator has finished an operation. Write 1 to clear, or write CTRL1 to clear.	0x0
		0	Busy or just cleared.	
		1	Completed last operation.	
3:1			Reserved.	
4	CARRY		Last carry value if operation produced a carry bit.	0x0
		0	Carry was 0 or no carry.	
		1	Carry was 1.	
5	BUSY		Indicates if the accelerator is busy performing an operation.	

Table 1060. Indicates operational status. (STATUS, offset 0xC) *…continued*

Bit	Symbol	Value	Description	Reset value
		0	Not busy and is idle.	
		1	Is busy.	
31:6			Reserved.	Undefined.

50.4.5 Interrupt set register

Sets the interrupts.

Table 1061. Sets interrupts (INTENSET, offset 0x10)

Bit	Symbol	Value	Description	Reset value
0	DONE		Set if the accelerator should interrupt when done.	0x0
		0	Do not interrupt when done.	
		1	Interrupt when done.	
31:1			Reserved.	

50.4.6 Interrupt clear register

Clears the interrupts.

Table 1062. Clears interrupts (INTENCLR, offset 0x14)

Bit	Symbol	Value	Description	Reset value
0	DONE		Written to clear an interrupt set with INTENSET.	0x0
		0	If written 0, ignored.	
		1	If written 1, do not Interrupt when done.	
31:1			Reserved.	

50.4.7 Interrupt status bits register

Provides status for interrupts.

Table 1063. Interrupt status bits (mask of INTENSET and STATUS) (INTSTAT, offset 0x18)

Bit	Symbol	Value	Description	Reset value
0	DONE		If set, interrupt is caused by accelerator being done.	0x0
		0	Not caused by accelerator being done.	
		1	Caused by accelerator being done.	
31:1			Reserved.	

50.4.8 Data (A-D) registers

Specifies register to be fed to multiplier.

Table 1064. Data registers A,B,C,D register (AREG, BREG, CREG, DREG, offset 0x20, 0x24, 0x28, 0x2C)

Bit	Symbol	Description	Reset value
31:0	REG_VALUE	Register to be fed into Multiplier. Is not normally written or read by application, but is available when accelerator not busy.	0x0

50.4.9 Result (0-3) registers

Holds working results for operation.

Table 1065. Result registers 0, 1, 2, 3 (RES0, RES1, RES2, RES3, offset 0x30, 0x34, 0x38, 0x3C)

Bit	Symbol	Description	Reset value
31:0	REG_VALUE	Register to hold working result (from multiplier, adder/xor, etc). Is not normally written or read by application, but is available when accelerator not busy.	ext

50.4.10 Mask register

Optional mask register used to apply a side channel countermeasure.

Table 1066. Mask register (MASK, offset 0x60))

Bit	Symbol	Description	Reset value
31:0	MASK	Mask to apply as side channel countermeasure. 0: No mask to be used. N: Mask to XOR onto values.	0x0

50.4.11 Re-mask register

Optional mask register used to apply a side channel countermeasure.

Table 1067. Re-mask register (REMASK, offset 0x64)

Bit	Symbol	Description	Reset value
31:0	REMASK	Mask to apply as side channel countermeasure. 0: No mask to be used. N: Mask to XOR onto values.	N/A

50.4.12 Security lock register

Reads back with security level locked to, or 0. Writes as 0 to unlock, 1 to lock.

Table 1068. Security lock register (LOCK, offset 0x80)

Bit	Symbol	Description	Reset value
0	LOCK	Reads back with security level locked to, or 0. Writes as 0 to unlock, 1 to lock.	0x0
31:1	-	Reserved.	0x0

51.1 How to read this chapter

Debugging is supported through Arm's Serial Wire Debug (SWD) interface, enabling debug with a range of debug probes and tools from NXP and other ecosystem partners. This chapter is intended for debug tool developers and assumes the reader has prior knowledge of Arm's SWD interface and Coresight debug and trace technology.

Note that JTAG is used on this device for boundary scan and production test only and cannot to used for debugging purposes, hence is not described in this chapter.

51.2 Features

- Supports arm serial wire debug mode for the CPU0 and CPU1.
- Trace port provides Cortex-M33 CPU instruction trace capability on CPU0 and CPU1. Output via a serial wire viewer.
- Direct debug access to all memories, registers, and peripherals.
- No target resources are required for the debugging session.
- Breakpoints: CPU0 and CPU1 includes eight instruction breakpoints.
- Watch-points: CPU0 and CPU1 includes four data watch-points that can also be used as triggers.
- Instrumentation Trace Macrocell allows additional software controlled trace for CPU0 and CPU1.

51.3 Basic configuration

This device supports Arm's Serial wire Debug (SWD) interface. SWD is the default function for pins PIO0_11 (SWCLK) and PIO0_12 (SWDIO) after a reset. If SWO is to be used, it must be enabled in the application code by selection the SWO function on either PIO0_8 or PIO0_10.

Debug by a remote host is controlled by the ROM and is only enabled when permitted through the device configuration, and when the correct protocol is followed to initiate a debug session. If the device has been configured for debug authentication, then a debug session must be initiated following the correct authentication sequence. See Section 51.6 for details of the mailbox protocol and Section 51.7 for details of debug authentication.

51.4 Pin description

Table 1069 indicates the various pin functions related to the debug process. Some of these functions share pins with other functions which therefore may not be used at the same time. Trace using the Serial Wire Output has limited bandwidth.

Table 1069.Serial Wire Debug pin description

Function	Type	Connect to	Reset value
SWCLK	In	PIO0_11	**Serial wire clock.** This pin is the clock for SWD debug logic when in the Serial Wire Debug mode (SWD). At reset release, this pin is pulled down internally. SWCLK is the default function of this pin.
SWDIO	I/O	PIO0_12	**Serial wire debug data input/output.** The SWDIO pin is used by an external debug tool to communicate with and control the part. At reset release, this pin is pulled up internally. SWDIO is the default function of this pin.
SWO	Out	PIO0_10 or PIO0_8	**Serial wire output.** The SWO pin optionally provides data from the ITM for an external debug tool to evaluate. SWO must be selected as a function on one of these pins prior to use.

The following setups are required to enable SWO output on GPIO PIO0_10 (FUNC6) or PIO0_8 (FUNC4):

1. Write 0x0 to TRACECLKDIV, see Section 4.5.48 “Trace clock divider register” to enable the trace clock divider.
2. If the clock to the IOCON block is not already enabled, write to AHBCLKCTRLSET0 Section 4.5.20 “AHB clock control set register 0”. The clock must be enabled in order to access any IOCON registers.
3. Set the FUNC6 value in the IOCON register corresponding to PIO0_10 or set the FUNC4 value in the IOCON register corresponding to PIO0_8.

51.5 Debug Subsystem functional description

This section describes the hardware elements of the Debug Subsystem and defines the command protocols used by them. See Section 51.7 and Section 51.6 respectively for descriptions of how the host debug system interacts with the Debug Subsystem during debug sessions with and without debug authentication.

51.5.1 Debug subsystem

Figure 199 shows the top-level debug ports and connections in the LPC55S6x/LPC55S2x/LPC552x. The designation AP is used to specify Access Port. Blocks SWJ-DP and DM-AP are always enabled and are accessible through the SWD interface. Remaining block are enabled/disabled under hardware state machine and software control.

- DAP: Debug access port which has Serial Wire port (SWJ-DP) which interprets the data coming in and routes to appropriate Access Port (AP).
- CPU0 AP: Debug access port for Cortex-M33 core instantiated as CPU0.
- CPU1 AP: Debug access port for Cortex-M33 core instantiated as CPU1. This instance of CM33 does not have security extension (TrustZone for Armv8-M).
- DM-AP: Debug Access port for debug mailbox. The Debug Mailbox is used to communicate with code executing from the ROM by sending/receiving messages.
  - This port is always enabled and the external world can send and receive data to/from ROM.
  - This port is used to implement NXP debug authentication protocol version 1.0.

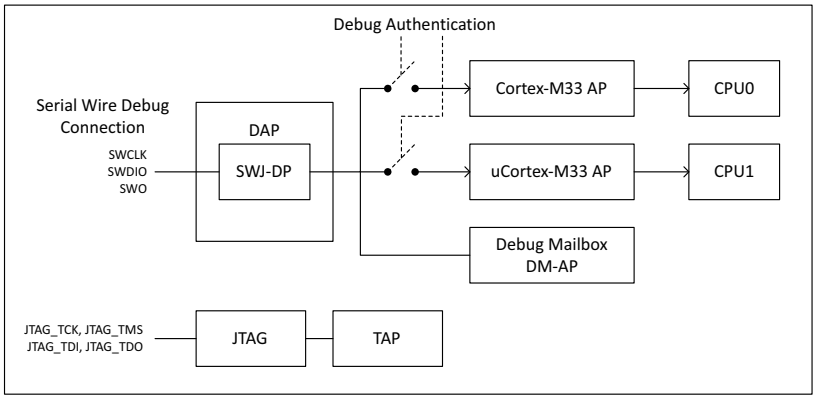


Fig 199. Internal debug system elements and connections

51.5.2 Debug Access Port (DAP)

The DAP has a Serial Wire Port (SWJ-DP) which interprets data coming in from a host debug system and routes it to the appropriate access port. External I/O pins that interface with the DAP are described in Table 1069. The DAP block is always enabled but the I/O pins that provide access to the SWD signals may be used for other functions under control of software (reference IOCON chapter).

51.5.3 CPU0 AP

The DAP for CPU0 is disabled during a power on reset or assertion of the reset pin and enabled by the ROM if/when the correct debug initiation procedure(s) are followed. If DAP is not being used, the debug enablement protocol can be used to initiate a debug session. If Debug Authentication is required, see Section 51.7 “Debug authentication”. The Debug authentication process allows control of the DBGEN, NIDEN, SPIDEN and SPNIDEN signals generated by the Cortex-M33 as described below.

DBGEN: Invasive debugging of TrustZone for Arm8-M defined non-secure domain.

- Breakpoints and watch points to halt the processor on specific activity.
- A debug connection to examine and modify registers and memory, and provide single-step execution.

NIDEN: Non-Invasive debugging of TrustZone for Arm8-M defined non-secure domain.

- A collection of information on instruction execution and data transfers.
- Deliver trace to off-chip in real-time to tools to merge data with source code on a development workstation for future analysis.

SPIDEN: Invasive debugging of TrustZone for Arm8-M defined secure domain.

SPNIDEN: Non-Invasive debugging of TrustZone for Arm8-M defined secure domain.

51.5.4 CPU1 AP

CPU1 is in reset mode by default and reset must be released by CPU0 to make CPU1 AP accessible. Debug Access port for CPU1 (Cortex-M33 core) is disabled on reset and enabled by HW state machine and through CODESECURITYPROTCPU1 register (offset 0xFB8 in SYSCON). This port has additional control to enable/disable different features as described below, controlled through DEBUG_FEATURES register (offset 0xFA4 in SYSCON) and DEBUG_FEATURES_DP register (offset 0xFA8 in SYSCON).

DBGEN: Invasive debugging of TrustZone for Arm8-M defined non-secure domain.

- Breakpoints and watch-points to halt the processor on specific activity.
- A debug connection to examine and modify registers and memory, and provide single-step execution.

NIDEN: Non-Invasive debugging of TrustZone for Arm8-M defined non-secure domain.

- A collection of information on instruction execution and data transfers.
- Deliver trace to off-chip in real-time to tools to merge data with source code on a development workstation for future analysis.

51.5.5 Debugger Mailbox AP

The Debugger Mailbox (DM) AP is a register based mailbox that is accessible by CPU0 and the device debug port DP of the MCU. This port is always enabled and an external host communicating via the SWD interface can exchange messages and data with the boot code executing from ROM on CPU0. This port is used to implement the NXP Debug Authentication Protocol. See Section 51.7 “Debug authentication” for a description of the protocol used to initiate a debug session from a host debug system.

51.5.5.1 Register description

The registers in the debug mailbox are shown in Table 1070. These registers are readable by the CPU and are intended primarily to allow on-chip ROM routines to implement requests from an external debugger.

Table 1070.Register overview: DBGMailbox (base address = 0x5009 C000)

Name	Access	Offset	Description	Reset value	Section
CSW	R/W	0x000	Command and status word.	0x0	51.5.5.1.1
REQUEST	R/W	0x004	Request from the debugger to the device.	0x0	51.5.5.1.2
RETURN	R/W	0x008	Return value from the device to the debugger.	0x0	51.5.5.1.3
ID	RO	0x0FC	Identification register.	0x002A 0000	51.5.5.1.4

51.5.5.1.1 Command and Status Word register

The CSW register contains command and status bits to facilitate communication between the debugger and the device.

Table 1071. Command and Status Word register (CSW, offset = 0x000)

Bit	Symbol	Description	Reset value
0	RESYNCH_REQ	The debugger sets this bit to request a re-synchronization.	0x0
1	REQ_PENDING	A request is pending for the debugger: a value is waiting to be read from the REQUEST register.	0x0
2	DBG_OR_ERR	When 1, a debug overrun has occurred: a REQUEST value has been overwritten by the debugger before it was read by the device.	0x0
3	AHB_OR_ERR	When 1, an AHB overrun has occurred: a RETURN value has been overwritten by the device before it was read by the debugger.	0x0
4	SOFT_RESET	Setting this write-only bit in the CSW register of the DM-AP by an external debugger resets the DM-AP state machine and its registers. While setting RESYNCH_REQ only resets the DP and CPU handshake state machine.	0x0
5	CHIP_RESET_REQ	This write-only bit causes the device (but not the DM-AP) to be reset by generating SYSRESET_REQ.	0x0
31:6		Reserved.	-

51.5.5.1.2 Request value register

The REQUEST register is used by a debugger to send action requests to the device.

Table 1072. Request value register (REQUEST, offset = 0x004)

Bit	Symbol	Description	Reset value
31:0	REQ	Request value. Reads as 0 when no new request is present. Cleared by the device. Can be read back by the debugger in order to confirm communication.	0x0

51.5.5.1.3 Return value register

The RETURN register provides any response from the device to the debugger.

Table 1073. Return value register (RETURN, offset = 0x008)

Bit	Symbol	Description	Reset value
31:0	RET	Return value. This is any response from the device to the debugger. If no new data is present, a debugger read will be stalled until new data is available.	0x0

51.5.5.1.4 Identification register

The ID register provides an identification of the DM-AP interface.

Table 1074. Identification register (ID, offset = 0x0FC)

Bit	Symbol	Description	Reset value
31:0	ID	Identification value.	0x002A0000

51.5.6 Reset handling

The debug domain (DP, CM33-AP, DM-AP) is reset upon POR (Power On Reset) or pin reset (assertion of nRESET). On other resets, the debug domain retains its state and the defined breakpoints, watch points, etc., survive even when the debugging tool issues a reset to the device. The matrix in Table 1075 describes the various reset types and shows which associated domains/components get reset by each type. “Rst” indicates that the sub-domain is reset by the associated reset source, and “Act” indicates that it remains active and does not change state in response to that reset source.

The DAP for CPU0 is disabled during a power on reset or assertion of the reset pin and enabled by the ROM if/when the correct debug initiation procedure(s) are followed. If the DAP is not being used, the debug enablement protocol can be used to initiate a debug session. If Debug Authentication is required, see Section 51.7 “Debug authentication”.

Table 1075. Resets

Chip reset/wakeup for the following domains and components:		POR (Power on Reset)	nRESET	BOD RESET	SYSTEMRESET	Debug mailbox	WDT RESET	SWR RESET	DPDRESET_WAKEUPIO	DPDRESET_RTC	DPDRESET_OSTIMER	Wakeup from Power Down	Wakeup from Deep Sleep	Wakeup from Sleepw
Always On Domain	PMC	Rst	Rst	Rst	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act
	OSTIMER	Rst	Rst	Rst	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act
	RTC	Rst	Rst	Rst	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act
System Domain	IOCON	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act	Act
	SYSCON	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act	Act
	SYSCON (debugger mailbox enable)	Rst	Rst	Rst	Act	Act	Act	Act	Rst	Rst	Rst	Act	Act	Act
	PUF key management	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act	Act
	GINT0/1	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act	Act
	flexcomm_3	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act	Act
Core Domain	SRAM Retention Memory	Rst	Rst	Rst	Act	Act	Act	Rst	Act	Act	Act	Act	Act	Act
	SRAM Memory	Rst	Rst	Rst	Act	Act	Act	Rst	Rst	Rst	Rst	Rst	Act	Act
	FLASH Memory	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	Flash controller	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	PRINCE	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act	Act
	WWDT	Rst	Rst	Rst	Rst	Rst	Act	Rst	Rst	Rst	Rst	Rst	Act	Act
	UTICK	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	MRT	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	SCT	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	Debug mailbox	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	cpu0	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act	Act
	CASPER	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	PowerQuad	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	PUF	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	RNG	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	CRC	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	HASH	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act

Table 1075. Resets

Chip reset/wakeup for the following domains and components:		POR (Power on Reset)	nRESET	BOD RESET	SYSTEMRESET	Debug mailbox	WDT RESET	SWR RESET	DPDRESET_WAKEUPIO	DPDRESET_RTC	DPDRESET_OSTIMER	Wakeup from Power Down	Wakeup from Deep Sleep	Wakeup from Sleepw
	Secure AHB ctrl	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act	Act
	DMA	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	analog_ctrl	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	ADC	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	USBFS	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	USBHS	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	ctimers (32bit)	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	SDIO (32bit)	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	flexcomm (all except 3)	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	INPUTMUX	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	GPIO	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	PINT	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	SECURE GPIO	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	SECURE PINT	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	PLU	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	i2s sharing	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
Analog Components	DCDC	Rst	Rst	Rst	Act	Act	Act	Rst	Rst	Rst	Rst	Rst	Act	Act
	Bias	Rst	Rst	Rst	Act	Act	Act	Rst	Rst	Rst	Rst	Rst	Act	Act
	BOD (VBAT)	Rst	Rst	Rst	Act	Act	Act	Rst	Rst	Rst	Rst	Rst	Act	Act
	BOD (CORE)	Rst	Rst	Rst	Act	Act	Act	Rst	Rst	Rst	Rst	Rst	Act	Act
	LDO_AO	Rst	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act
	LDOs	Rst	Rst	Rst	Act	Act	Act	Rst	Rst	Rst	Rst	Rst	Act	Act
	32 kHz XTAL	Rst	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act
	32 MHz XTAL	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	32 kHz FRO	Rst	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act
	192 MHz FRO	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	1 MHz FRO	Rst	Rst	Rst	Act	Act	Act	Rst	Rst	Rst	Rst	Rst	Act	Act
	Temperature sensor	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	ADC	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	Analog Comparator	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	USB HS Phy	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act

51.5.7 Mailbox commands

This section describes the Response and Request message formats and available mailbox commands and their associated parameters.

51.5.7.1 Request

The first word transmitted in a request is a header word containing the command ID and number of following data words. The command packet is sent to the device by writing 32-bits at a time to the REQUEST register. When sending command packets greater than 32-bits, the debugger should read an ACK_TOKEN in the RETURN register before writing the next 32-bits.

Following the header are the number of 32-bit words specified in the header.

Table 1076. Request register byte description

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	commandID[7:0]	commandID[15:8]	dataWordCount[7:0]	dataWordCount[15:8]
1	*data…*			

The C structure definition for a request is as follows:

```
struct dm_request {
      uint16_t commandID;
      uint16_t dataWordCount;
      uint32_t data[];
};
```

51.5.7.1.1 DM-AP commands

Commands for the DM-AP are listed below. These are written to the REQUEST register. With the exception of the “Enter ISP mode' and 'Start Debug Session” command, the issuance of a command (or sequence of commands) is normally followed by the issuance of an “Exit DM_AP” command to resume normal device boot flow.

Table 1077. DM-AP commands

Name	Command code	Parameters	Response	Description
Start DM-AP (legacy command)	0x01	None	32-bit status	Causes the device to enter DM-AP command mode prior to sending Bulk Erase commands. This command is provided for backwards compatibility and does not need to be used.
Reserved	0x02	None	32-bit status	Reserved, Returns 3.
Bulk Erase (legacy command)	0x03	None	32-bit status	Erase the entire on-chip flash memory (excluding Protected Flash Regions).
Exit DM_AP (legacy command)	0x04	None	32-bit status	Causes the device to continue normal debug flow.

Table 1077. DM-AP commands *...continued*

Name	Command code	Parameters	Response	Description
Enter ISP Mode	0x05	dataWordCount: 0x1 data[0]: ISP mode enum 0xffffffff - Auto detection 0x1 - UART 0x2 - I2C 0x4 - SPI 0x10 - USB-HID Others - Reserved	32-bit status	Enter specified ISP mode. By default, ISP mode entry is determined by the state of the ISP boot selection pins at reset time. Usually this functionality is disabled through PFR configuration prior to field deployment. This command can be used to enter ISP mode in those situation or when ISP boot selection pins are used for some other board function. Support of this command can be restricted through the DCFG_SOCU field in the PFR, as described in Section 51.7 “Debug authentication”.
Set FA Mode	0x06	None	32-bit status	Sets the part permanently in "Fault Analysis" mode for return to the NXP factory. Upon receiving this command boot code in ROM, customer sensitive assets (key codes) stored in PFR are erased. In addition, the Fault Analysis (FA) or RMA mode bit in the PFR field are set so suspect parts can be sent to NXP for FA/RMA testing. Support of this command can be restricted through the DCFG_SOCU field in the PFR, as described in Section 51.7 “Debug authentication”.
Start Debug Session	0x07	None	32-bit status	When program control is in ROM memory context (i.e., instructions are fetched from a ROM memory address range) during boot, the debug access is disabled irrespective of the device life-cycle state or DCFG_SOCU settings. This command instructs the boot code in ROM to clean-up memory and peripherals (SysTick, UART, SPI, I2C, QSPI, SD/MMC, USB, MPU and SAU) initialized by boot code and to enter safe processor execution context for external debuggers to attach. On systems, when there is no valid image in boot media, the program control enters ISP command handler loop, which is still in ROM memory context, and hence debug access is disabled. As a result, an external debug system has to use this command to indicate to the ROM its intention of connecting to the debugger. Upon receiving the command, the ROM cleans all peripheral configurations and secrets before enabling debug access. After enabling debug access, the ROM enters a while(1) loop.
Debug Auth. Start	0x10	None	dataWordCount: 0x12C or 0x22C data[]: DAC	Start Debug Authentication Protocol. ROM responds to debugger with *DAC (Debug Authentication Challenge)* message
Debug Auth. Response	0x11	dataWordCount: 0x12C data[]: DAR	32-bit status	Debug Authentication Response

51.5.7.2 Response

The first word transmitted in a response is a header word containing the command status and number of following data words. The command response can be read 32-bits at a time through the RETURN register. The initial 32-bits contains the response header as shown in Table 1078. When reading a response greater than 32-bits, the debugger should write the ACK_TOKEN to the REQUEST register after every read of the RETURN register until the full response packet is received.

- To support legacy LPC commands and response values, Bit 31 in the header is used to indicate that the response follows a new protocol structure (this bit is set when the new protocol is used).

Table 1078. Response register byte description

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	bits[7:0]:cmdStatus[7:0]	bits[7:0]:cmdStatus[15:8]	bits[7:0]:dataWordCount[7:0]	bits[6:0]:dataWordCount[14:8] bit[7]: new_protocol
1	*data…*			

The C structure definition for a response is as follows:

```
struct dm_response {
     uint16_t commandStatus;
     uint16_t dataWordCount;
     uint32_t data[];
};
```

51.5.7.2.1 Response codes for DM-AP commands

The response codes for DM-AP commands are listed below. These commands can be read from the RETURN register (see Table 1079). The upper 15-bits (bits 30:16) of the response code only contains relevant data (i.e., the data word count related to the command) when the command is successful.

Table 1079. DM-AP response codes

Return code (bits 15:0)	
0x0000	Command succeeded.
0x0001	Debug mode not entered. This is returned if other commands are sent prior to the “Enter DM-AP” command.
0x0002	Command not recognized. A command was received other than the ones defined above.
0x0003	Command failed.

51.5.7.3 ACK_TOKEN

- When a command has parameters, the debugger waits for the ACK_TOKEN (sent through the RETURN register) before sending the next 32-bit value.
- When the device has a response packet to send back to the debugger, the ROM code waits for the debugger to send an ACK_TOKEN (sent through the REQUEST register) before sending the next 32-bit value.
- The upper 16-bits of ACK_TOKEN are set by the receiving end with the number of remaining words that are expected.
- Lower 16-bits are always set to 0xA5A5.

Table 1080. ACK_TOKEN register byte description

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0xA5	0xA5	remainCount[7:0]	remainCount[15:8]

The C structure definition for a ACK_TOKEN is as follows:

```
struct dm_ack_token {
     uint16_t token;   /* always set to 0xA5A5 */
     uint16_t remainCount;  /* count of remaining word */
};
```

51.5.7.4 Error handling

If an overrun occurs from either side of the communication, the appropriate error flag is set in the CSW. The state machine hardware will prevent further communication in any direction once an overrun has occurred in that direction, so if such an error occurs, the debugger will need to start with a new re-synchronization request in order to clear the error flag.

51.6 Debug session protocol

LPC55S6x/LPC55S2x Boot ROM implements debug mailbox protocol to interact with host debug systems over the SWD interface.

The protocol has following features:

- Request/response based.
- Support for relatively large command and response data.
- All commands and responses are 32-bit word aligned.
- Supports data above 32-bits by using an ACK_TOKEN that moderates the transfer in 32-bit value chunks.
- Requests and responses use the same basic structure.

The ROM provides three debug methods/mechanisms to attach the debugger in a predictable manner:

- Start debug session method
- Debug access trigger method
- Debug authentication mechanism

Debug authentication is disabled by default, and is set up (if required) during development or device provisioning during the production phase of a product using the device. See Section 51.7 for full details of debug authentication.

On LPC55S6x/LPC55S2x parts, when program control is in ROM memory context (i.e., instructions are fetched from the ROM memory address range) during the boot process, the debug access port (AP) of CPU0 is disabled irrespective of device life-cycle state or DCFG_SOCU settings. This mechanism is referred as 'Boot-ROM protection' in this document. Thus the method to initiate a debug session will vary depending on the device state and intended debug scenario. The scenarios described in the rest of these section are as follows:

- Flash is uninitialized (as with new part from the NXP factory) or does not contain a valid, bootable image. If the flash does not contain a valid image, the ROM will proceed to ISP mode and wait to be booted via one of its serial interfaces; in ISP mode the debug interface is disabled.
- ISP mode, initiated because the ISP pin was asserted on the device at reset.
- Connection to a device running a valid application, with the intent to update flash with a new application.
- Connection to a device running a valid application in flash, with the intent to debug without updating flash (also called a “debug attach”).

51.6.1 Debug session with uninitialized/invalid flash image or ISP mode

When the device boots, there may be no valid image in the boot media, at which point the ROM-based program control enters the In-System Programing (ISP) loop, and debug access is disabled for security reasons. Another scenario is where the device may be placed into ISP mode because the ISP has been asserted as the device leaves reset. This section describes how to establish a debug session for these scenarios.

To ensure the state machine controlling debug mailbox commands is in a known state, the debugger may need to reset this logic; this is done by setting the RESYNCH_REQ bit in the CSW register. The debugger must then reset the device by either writing a 1 to the CHIP_RESET_REQ bit in the CSW. After requesting a re-synchronization and resetting the device, the debugger reads the CSW register. The debugger must wait until the device has completed the re-synchronization process, indicated by reading a value of 0 from the CSW register (checking only the lower 16-bits of this 32-bit value).

To start a debug session and control the exchange of debug information, use the DM-AP commands. Following a successful initial re-synchronization, communication by the debugger to the device is achieved using 32-bit DM-AP command writes to the REQUEST register in the Debug Mailbox (DM-AP Commands are shown in Table 1077.) The debugger can read results of communications via the RETURN register. The debugger should poll the RETURN register in the same manner as it polled the CSW following a re-synchronization request in order to ensure transactions have completed successfully. Response codes are shown in Table 1079. To handle situations where debug is disabled due to Boot-ROM protection, the debug system must use a specific command (Start Debug Session) and follow the debug mailbox protocol to indicate to the ROM-based boot code its intention of connecting a debug session over SWD. Upon receiving the command, the boot code ensures any unwanted peripheral interrupts are disabled and secrets managed before enabling debug access. After enabling debug access, the ROM enters a while (1) loop.

Once the Start Debug Session and device reset have been successfully executed, the AP for Core0 will be accessible and can used to set breakpoints, etc. as with other Cortex-M devices.

Following is a sample that shows how a debug session is initiated for the scenarios described above:

```
// Pseudo Code Syntax
// ------------------------------------------------------------------------
// WriteDP  <register> <value>
// value = ReadDP  <register>
```

```
// AP transactions presume the DM AP is selected
// WriteAP <register> <value>
// value = ReadAP  <register> <value>
// -------------------------------------------------------------------------

// Read AP ID register to identify DM AP at index 2
WriteDP 2 0x020000F0
// The returned AP ID should be 0x002A0000
value = ReadAP 3
print "AP ID: ", value



// Select DM AP index 2
WriteDP 2 0x02000000

// Write DM RESYNC_REQ + CHIP_RESET_REQ
WriteAP 0 0x21

// Poll CSW register (0) for zero return, indicating success
value = -1
while  value != 0 {
     value = ReadAP 0
}
print "RESYNC_REQ + CHIP_RESET_REQ: ", value

// Write DM START_DBG_SESSION to REQUEST register (1)
WriteAP 1 7

// Poll RETURN register (2) for zero return
value = -1
while  value != 0 {
     value = (ReadAP 2 & 0xFFFF)
}
print "DEBUG_SESSION_REQ: ", value
```

Following a successful debug connection, a flash loader would normally be loaded into RAM and used to program the application to be debugged, and the required breakpoints in the application code set. Once this is done, a SYSRESET_REQ command should be issued to ensure the ROM fully executes (as would happen in a deployed end application) before reaching the downloaded application.

Note: The method described here for initiating a debug session is designed for current 1B silicon revisions of the part and will result in an endless loop when used on older 0A parts. Please errata sheet for further details on how to determine the version of the part and perform a workaround for initiating the debugging on an 0A part.

51.6.2 Debug session with valid application in flash

In this scenario description it is assumed that debug has not been disabled by an application already in flash, so the Core0 AP is visible and breakpoints can be set without the need to re-synchronize the Mailbox hardware or issue a Debug Session Request. The same methods described in Section 51.6.1 may be used in order to simplify debug support implementations.

51.6.3 Debug session attaching to a running target

This scenario is when the device has booted and is running an application that has not disabled debug, and the host system is attempting to connect to that device without reseting it and with no intention to update flash. In this case, the Core0 AP should be visible and breakpoints can be set without the need to re-synchronize the Mailbox hardware, issue a Debug Session Request or issue a reset.

51.6.4 Halting execution immediately following ROM execution

Traditionally, debug systems may set a vector catch at the reset vector in order to break code execution at this point, however the LPC55S6x/LPC55S2x/LPC552x ROM prevents this. To allow the debug system to halt execution immediately after the ROM has completed preparations after a debug session request, a watch point may be set to trigger on a read access to address 0x50000040.

51.7 Debug authentication

The fundamental principles of debugging, which require access to the system state and system information, conflict with the principles of security, which require the restriction of access to assets. Thus, many products disable debug access completely before deploying the product. This causes challenges for product design teams to do proper Return Material Analysis (RMA). To address these challenges, the LPC55S6x/LPC55S2x offers a debug authentication protocol as a mechanism to authenticate the debugger (an external entity) has the credentials approved by the product manufacturer before granting debug access to the device.

The debug authentication scheme on LPC55S6x/LPC55S2x is a challenge-response scheme and assures that debugger in possession of required debug credentials only can successfully authenticate over debug interface and access restricted parts of the device. This protocol is divided into steps as shown in Figure 200 and described below:

1. The debugger initiates the Debug Mailbox message exchange by setting the RESYNCH_REQ bit and CHIP_RESET_REQ bit in the CSW register of DM-AP.
2. The debugger waits (minimum 30 ms) for the devices to restart and enter debug mailbox request handling loop.
3. The debugger sends Debug Authentication Start command (command code 0x10) to the device.
4. The device responds back with Debug Authentication Challenge (DAC) packet based on the debug access rights pre-configured in CMPA fields, which are collectively referred as Device Credential Constraints Configuration (DCFG_CC). The response packet also contains a 32 bytes random challenge vector.

5. The debugger responds to the challenge with a Debug Authentication Response (DAR) message by using an appropriate debug certificate, matching the device identifier in the DAC. The DAR packet contains the debug access permission certificate, also referred as Debug Credential (DC), and a cryptographic signature binding the DC and the challenge vector provided in the DAC.
6. The device on receiving the DAR, validates the contents by verifying the cryptographic signature of the message using the debugger's public key present in the embedded the Debug Credential (DC). On successful validation of DAR, the device enables access to the debug domains permitted in the DC.

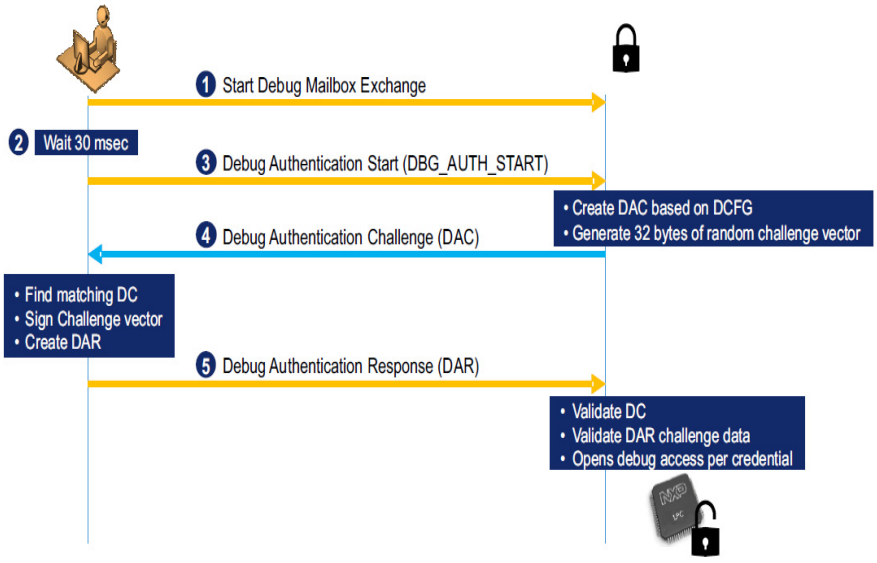


Fig 200. Debug authentication flow

51.7.1 Debug Access Control Configuration

The boot code present in LPC55S6x/LPC55S2x ROM handles the device side of Debug authentication process. However, the debug access control rights and security policies can be configured by programming the following configuration fields which are collectively referred to as Device Configuration for Credential Constraints (DCFG_CC), present in Customer Manufacturing Programmable Area (CMPA) and Customer Field Programmable Area (CFPA).

- DCFG_VER: This field controls the cryptographic primitives used during authentication.
- DCFG_ROTID: This field defines the Root of trust identifier (ROTID). The ROTID field is used to bind the devices to specific Certificate Authority (CA) keys issuing the debug credentials. These CA keys are also referred as Root of Trust (RoTK) keys.

- DCFG_UUID: Controls whether to enforce UUID check during Debug Credential (DC) validation. If this field is set only DC with matching device UUID can unlock the debug access.
- DCFG_CC_SOCU: This configuration field specifies access rights to various debug domains.
- DCFG_VENDOR_USAGE: This field can be used to define vendor specific debug policy use case such as DC revocations or department identifier. It is recommended to use field for revocation of already issued debug certificates.

These fields should be programmed as part of the OEM provisioning process.

51.7.1.1 Protocol Version (DCFG_VER)

The LPC55S6x/LPC55S2x supports two instantiations of Debug authentication protocol versions, which are defined based on the different-sized RSA keys.

- Version 1.0: Uses RSASSA-PKCS1-v1_5 signature verification using RSA keys with 2048-bit modulus and a 32-bit exponent.
- Version 1.1: Uses RSASSA-PKCS1-v1_5 signature verification using RSA keys with 4096-bit modulus and a 32-bit exponent.

To enforce usage of RSA4096 keys, set the RSA4K field in the SECURE_BOOT_CFG word (0x9E41C) to 0x2 in CMPA.

Note, both debug authentication certificates and image signing certificates use same Root of Trust keys (RoTK). Hence when this field is set the secure boot image signing key certificate chain should also use RSA 4096-bit keys.

51.7.1.2 Root of Trust Identifier (DCFG_ROTID)

The Root of Trust Identifier used in debug authentication protocol is composed of two elements.

1. A 256-bit cryptographic hash (SHA256) over the Root of Trust Keys Table. This is same as Root Keys Table Hash (RKTH) field referred in Secure boot ROM chapter. See: Chapter 7 “LPC55S6x/LPC55S2x/LPC552x Secure Boot ROM”. RKTH is a 32-byte SHA-256 hash of SHA-256 hashes of up to four root public keys.
2. A 32-bit field containing revocation bits for the four Root of Trust keys in the table.

The Root of Trust Identifier used in the debug authentication protocol is composed of two elements:

- CMPA words 0x9E450 – 0x9E46C specifies the ROTID.
- CFPA word ROTKH_REVOKE (at offset 0x18).

51.7.1.3 Enforce UUID checking (DCFG_UUID)

Controls whether to enforce UUID check during Debug Credential (DC) validation. If this field is set then only DC containing a UUID attribute that is an exact match to the device can unlock the debug access.

This field can be set by programming UUID_CHECK (bit 15) in CC_SOCU_PIN word (PFR address 0x9E410) in CMPA field or same bit position 15 in DCFG_CC_SOCU_NS_PIN word in CFPA pages.

This device-specific constraint, if enabled, is in addition to all the other constraints defined and enforced by the authentication protocol

51.7.1.4 Credential Constraints (DCFG_CC_SOCU)

The DCFG_CC_SOCU is a configuration that specifies debug access restrictions per debug domain. These access restrictions are also referred as constraint attributes in this section. The debug subsystem is sub-divided in to multiple debug domains to allow finer access control. Table 1082 “CC_LIST_Table” shows debug domains and their corresponding control bit position in DCFG_CC_SOCU. Which is logically composed of two components:

- SOCU_PIN: A bitmask that specifies which debug domains are predetermined by device configuration.
- SOCU_DFLT: Provides the final access level for those bits that the SOCU_PIN field indicated are predetermined by device configuration.

The following table shows the restriction levels:

Table 1081.Access restriction levels

Restriction Level	SOCU_PIN [n]	SOCU_DFLT [n]	Description
0	1	1	Access to the sub-domain is always enabled. This setting is provided for module use case scenario where DCFG_CC_SOCU_NS would be used to define further access restrictions before final deployment of the product. See Section 51.7.6.2 “Module use case with OEM tier1 and teir2 Lifecycle states” for details.
1	0	0	Access to the sub-domain is disabled at startup. But the access can be enabled through debug authentication process by providing appropriate Debug Credential (DC) certificate. Note: The LPC55S6x/LPC55S2x/LPC552x (non-S parts without security features) does not support debug authentication process. Hence this option is not available, but other options can be used to permanently enable/disable debug access on those devices.
2	0	1	Illegal setting. Part may lock-up if this setting is selected.
3	1	0	Access to the sub-domain is permanently disabled and can’t be reversed. This setting offers the highest level of restriction.

Debug domains supported by the LPC55S6x/LPC55S2x/LPC552x are shown in Table 1082.

Table 1082.CC_LIST_Table

Bit	Symbol	Description
0	NIDEN	Controls non-Invasive debugging of TrustZone for Arm8-M defined non-secure domain of CPU0.
1	DBGEN	Controls invasive debugging of TrustZone for Arm8-M defined non-secure domain of CPU0.
2	SPNIDEN	Controls non-Invasive debugging of TrustZone for Arm8-M defined secure domain of CPU0.
3	SPIDEN	Controls invasive debugging of TrustZone for Arm8-M defined secure domain of CPU0.
4	TAPEN	Controls TAP (Test Access Point) controller used for structural integrity testing of silicon by NXP as part of Return Material Analysis (RMA).
5	CPU1_DBGEN	Controls invasive debugging of CPU1 (valid for dual-core).

Table 1082.CC_LIST_Table *...continued*

Bit	Symbol	Description
6	ISP_CMD_EN	Controls whether ISP boot flow DM-AP command (command code: 0x05) can be issued after authentication.
7	FA_CMD_EN	Controls whether permanent modification DM-AP commands such as Bulk Erase (command code: 0x02) and Set FA Mode (command code: 0x06), can be issued through after authentication.
8	RSRVD	Reserved.
9	CPU1_NIDEN	Controls non-invasive debugging of CPU1 (valid for dual-core).
31:10	-	Reserved.

Table 1083.Layout of CC_SOCU_PIN (CMPA offset 0x10) & CC_SOCU_PIN_NS (CFPA offset 0x20)

Bits	Symbol	Description
14:0	SOCU_PIN[n]	Defines whether the restriction level for the sub-domains is fixed or controlled by debug authentication process. The bit encoding of this field is defined as per Table 1082 “CC_LIST_Table”.
15	UUID_CHECK	Controls whether to enforce UUID check during Debug Credential (DC) validation. If this bit is set then the device will only accept Debug Credential (DC) certificate containing UUID attribute that is an exact match to the device’s UUID. If a bit is set, access is allowed. Otherwise access is denied.
31:16	INV_SOCU_PIN[n]	Inverse value of the above bitfield [15:0].

Table 1084.Layout of CC_SOCU_DFLT (CMPA offset 0x14) & CC_SOCU_DFLT_NS (CFPA offset 0x24)

Bits	Symbol	Description
15:0	SOCU_DFLT[n]	Defines the restriction level for the sub-domains which are configured as predetermined in SOCU_PIN[n] field. The bit encoding of this field is defined as per Table 1082 “CC_LIST_Table”.
31:16	INV_SOCU_DFLT[n]	Inverse value of the above bitfield [15:0].

51.7.1.5 DCFG_VENDOR_USAGE

This field can be used to define vendor specific debug policy use case such as Debug Credential (DC) certificate revocations or department identifier or model identifier. During Debug Authentication Response (DAR) processing the device checks that the value specified in Vendor Usage field of DC matches exactly the value programmed in DCFG_VENDOR_USAGE fields of device configurations.

The LPC55S6x/LPC55S2x provides 4 bytes length DCFG_VENDOR_USAGE field, composed from following fields:

- Upper 2 bytes from CMPA.VENDOR_USAGE (PFR address offset 0x18)
  - Recommended to use this field to identify department or model number. So that Debug Credential (DC) Certificates can be issued on class basis instead of issuing UUID specific certificates.
- Lower 2 bytes from CFPA.VENDOR_USAGE (CFPA offset 0x01C).
  - In CFPA, this filed is implemented as a monotonic counter field. Whenever a new version of CFPA page is written this field can have same value or a higher value.
  - It is recommended to use this field to revoke DC certificates issued till that point.

51.7.2 Debug Credential Certificate (DC)

By prior construction, the debugger should already have a DCK (Debug Credential Key). The public key part of this key pair is used to represent the identity of the debugger through the creation of a DC, which binds that public key to usage attributes, and is then authorized/signed by the vendor's RoT key.

Total DC size is 940 bytes (v1.0) or 1708 bytes (v1.1).

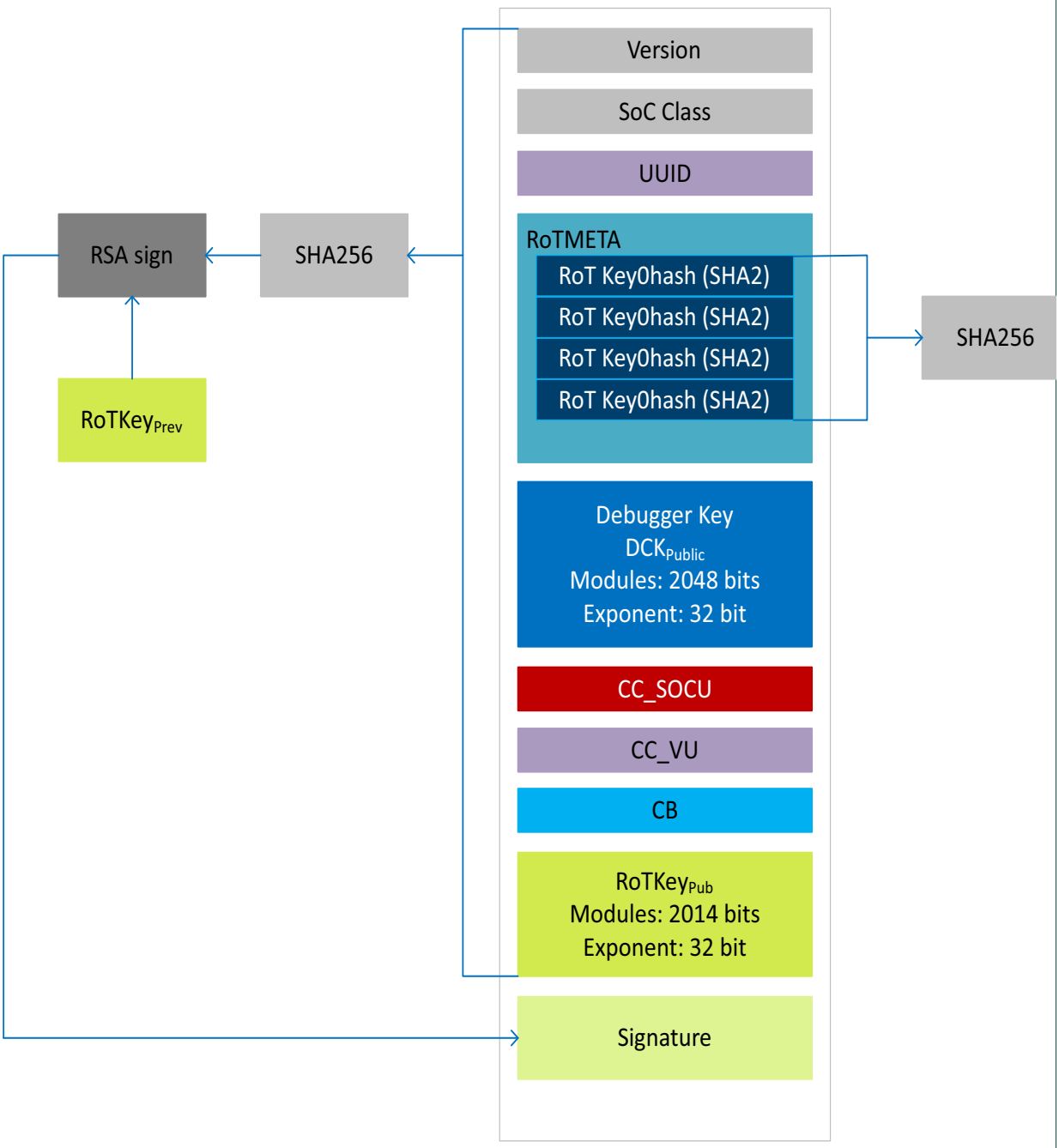


Fig 201. Fig 193.Debug Credential certificate fields

The data structure is a represented as a packed binary concatenation of its component fields as shown in the list below:

Table 1085.Debug Credential Certificate fields

Name	Offset	Description	Size in bytes
VERSION	0x000	Identifies the Debug Authentication protocol version. • Set 0x00010000 for v1.0, which uses RSA2048 keys • Set 0x00010001 for v1.1, which uses RSA4096 keys	4
SOCC	0x004	SoC class specifier. Always set this field to 0x0000 0001.	4
UUID	0x008	Unique device identifier, in case this certificate is used on a device configured for UUID-matching. If UUID matching is enabled certificate is restricted to a specific device, otherwise certificate is enabled for whole SoC Class.	16
ROTMETA	0x018	Meta data used for authenticating Certificate Authority key (a.k.a. RoT key) used for signing this certificate. The SHA256 hashes of four RoT public keys should be set in this field. Note, on this device same RoT keys are used for certifying image signing keys and debug keys.	128
DCK_MOD	0x098	Modulus value of Debugger public key (DCKpub). • For version v1.0, the modulus is 2048-bit (256 bytes). • For version v1.1, the modulus is 4096-bit (512 bytes).	256 or 512
DCK_EXP	0x198 or 0x298	The 32-bit exponent value of Debugger public key (DCKpub).	4
CC_SOCU	0x19C or 0x29C	SoC specific Credential Constraints. Specifies the debug access rights allowed for this certificate holder.	4
CC_VU	0x1A0 or 0x2A0	Vendor Usage. Should match DCFG_VENDOR_USAGE field in Device Configuration for Credential Constraints (DCFG_CC). See Section 51.7.1.5 "DCFG_VENDOR_USAGE". It can be used to revoke Debug Certificates.	4
CB	0x1A4 or 0x2A4	Credential beacon that vendor has associated with this DC. Only lower 16-bits of this field are effective. This field can be used to extend the Debug Authentication process. When a non-zero value is used in this field ROM differs opening debug access to user application. The result of the authentication process is written to DBG_FEATURES register while the user application after doing its extended processing, such as clean-up of critical keys & Secrets, should copy the value to DBG_FEATURES_DP register to enable the debug access. To aid user application ROM stores beacon values in DEBUG_AUTH_BEACON register (0x40000FC0)	4
RoTK_MOD	0x1A8 or 0x2A8	Modulus value of Root of Trust Vendor public key used for signing this certificate. • For version v1.0, the modulus is 2048-bit (256 bytes). • For version v1.1, the modulus is 4096-bit (512 bytes).	256 or 512
RoTK_EXP	0x2A8 or 0x4A8	The 32-bit exponent value of Root of Trust Vendor public key used for signing this certificate.	4
SIG	0x2AC or 0x4AC	A cryptographic signature by the RoT over the eight previous fields. This ensures the DC is "blessed" by the Vendor for use by the debugger. • 256 bytes (v1.0) or 512 bytes (v1.1), as calculated below. • SIG(RoTpriv,HASH(DC :: 1II DC :: 2II ... IIDC :: 8)) • SHA256 is used for Hash function. • RSASSA-PKCS1-v1_5 is used for SIG function. • RSA signature scheme from PKCS1 specification v1.5 (RFC 2313).	256 or 512

51.7.3 Debug Authentication Challenge (DAC)

The debug authentication protocol begins with a DAC (Debug Authentication Challenge) message, issued by the device to the debugger. Total message size is 104 bytes.

From debug authentication protocol perspective, what matters is that the debugger selects a Debug Credential (DC) certificate that will successfully authenticate, based on the constraints provided in the DAC, and will provide the required debug behavior post-authentication (for example, whether to debug secure world, with the desired credential beacon). If such a credential cannot be found, the debugger should report a corresponding error to the user.

Note: The debugger must also be able to produce signatures using the private key corresponding to the selected DC, so that any credential store can manage this association between credentials and the corresponding private keys.

The named elements of this message are shown in Figure 202.

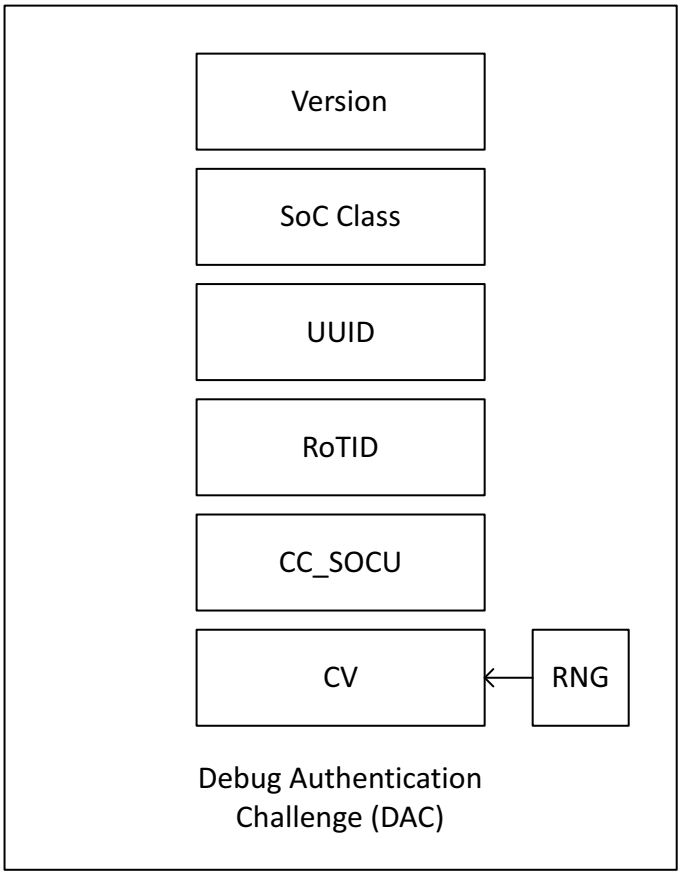


Fig 202. Debug Authentication Challenge (DAC) fields

Table 1086.Debug Authentication Challenge (DAC) fields

Name	Offset	Description	Size in bytes
VERSION	0x000	Identifies the Debug Authentication protocol version. The value returned in this field is set based on the RSA4K field is set to • 0x00010000 indicating protocol version v1.0, when DCFG_RSA4K is set to 0. • 0x00010001 for v1.1, which uses RSA4096 keys.	4
SOCC	0x004	SoC class specifier. This field is always set to 0x0000 0001.	4
UUID	0x008	Unique device identifier. If UUID matching is enabled in DCFG_CC_SoCU, then device will include its UUID in the challenge packet. Or else this field is set to zeros.	16
ROTID	0x018	Meta data used for authenticating Certificate Authority key (a.k.a. RoT key) used for signing DC certificate. SHA256 hashes of four RoT public keys would be set in this field. Check Section 51.7.1.2 “Root of Trust Identifier (DCFG_ROTID)” for details. Note, on this device same RoT keys are used for certifying image signing keys and debug keys. • 32 bytes RKTH value. • 4 bytes containing revocation flags. Only least significant 4 bits are used.	36
CC	0x3C	Credential Constraints. Specifies the debug access rights allowed for this certificate holder. • A 32-bit SOCU_PIN value. See Table 1082 “CC_LIST_Table” for bit encoding of this field. • A 32-bit SOCU_DFLT value. See Table 1082 “CC_LIST_Table” for bit encoding of this field. • A 32-bit DCFG_VENDOR_USAGE value.	12
CV	0x48	Challenge Vector generated by the device. Device provides 32 bytes random value generated using TRNG block.	32

51.7.4 Debug Authentication Response (DAR)

Before the debugger can formulate a response to the challenge, it should perform some checks to confirm correctness of VER, SoCC, UUID, RoTID and CC; it should find a matching DC.

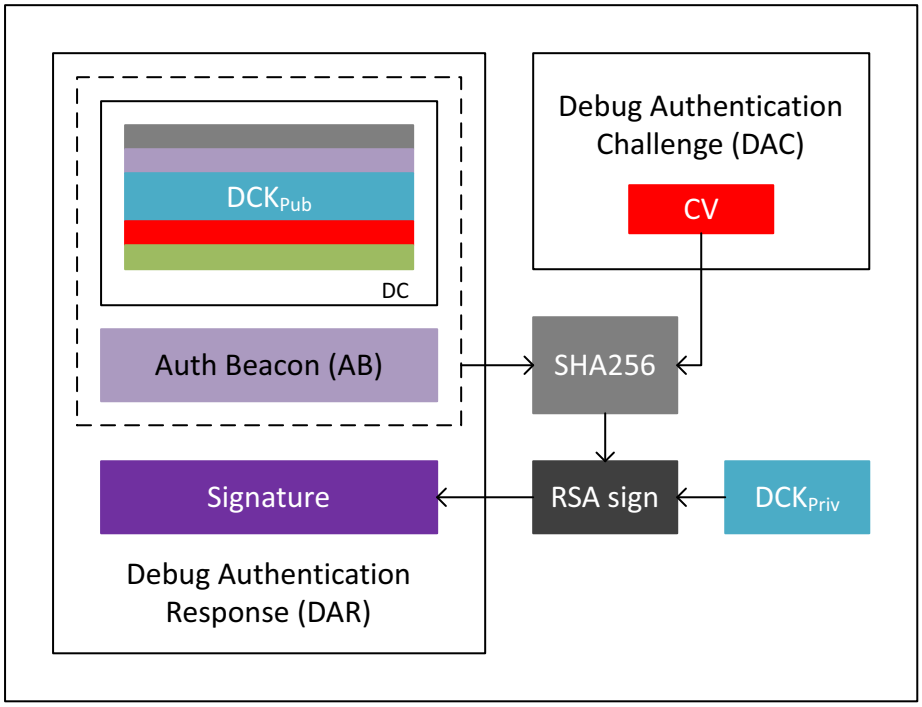


Fig 203. Debug Authentication Response (DAR) fields

Table 1087.Debug Authentication Response (DAR) fields

Name	Offset	Description	Size in bytes
DC	0x000	DC, provides the debugger's credential, RoT public key and more, described in Debug Credential (Certificate). See Section 51.7.2 "Debug Credential Certificate (DC)". • 940 bytes (v1.0) or • 1708 bytes (v1.1)	940 or 1708
AB	0x3AC or 0x6AC	AB, the Authentication Beacon provided and signed by the debugger during the authentication process. Refer to the Credential Beacon (CB) field in Section 51.7.2 "Debug Credential Certificate (DC)" structure for details.	4
SIG	0x3B0 or 0x6B0	A cryptographic signature by the debugger that binds the previous two fields with the challenge vector from the DAC. SIG = RSA_SIGN (DCKpriv,SHA256(DAR::DC II DAR::AB II DAR::CV)) • Uses the private key corresponding to the public key (DCK) of the selected DC (proves that debugger has possession of debugger private key). • RSASSA-PKCS1-v1_5 is used for SIG function.	256 or 512

51.7.5 Device processing the DAR

The device Boot ROM will process DAR received from debugger. As a part of the validation step, device will:

- Verify DC: Validate DC version, SoCC, UUID, RoT, VU, and DC signature.
- Verify that the DAR has a valid signature that binds it to the CV from the DAC.

If all the steps are successfully completed, it can be deduced that:

- The debugger possesses the private key corresponding to the vendor/RoT-signed credential.
- The credential satisfies the constraints specified in the device configuration.
- The response of the debugger to the challenge from the device is produced and signed in response to the challenge (because of its cryptographic dependency on the challenge vector). The response is not replayed from a previous authentication where a different challenge vector is used.

After completion of processing DAR, if authentication is successful, Debug Access will be granted. If authentication fails, no special response is issued but further debug access request will be ignored, and device will enter in a failure loop.

51.7.5.1 Successful authentication

ROM executes following steps upon successful debug authentication:

1. ROM determines the final enable states of the debug ports based on pinned state from DCFG_CC_SOCU and the DC::CC fields.
2. ROM evaluates part enables using following logic:
   - Uses pinned states based on DCFG_CC_SOCU and DCFG_CC_SOCU_NS words.
   - Evaluate SOCU_PIN and SOCU_DFLT
   - Evaluates debug port enables for ports which are not pinned using authentication protocol.
   - Debug_State = (SOCU_PIN & SOCU_DFLT) | (! SOCU_PIN & DC:: CC_SOCU)
   - Enables debug ports for bits which are set in above evaluation.
3. In Debug Mailbox handler allows following commands only if the enable bit is set in final evaluation of Debug_State.
   - Handle 'ENTER_ISP_MODE' command only if default ISP_CMD_EN bit is set in Debug_State.
   - Handles 'SET_FA_MODE' and 'ERASE_FLASH' commands only if default FA_CMD_EN bit is set in Debug_State.
4. ROM stores the beacons in the write lockable register DBG_AUTH_SCRATCH.
   - DBG_AUTH_SCRATCH [15:0] = DAR::DC::CB[15:0]
   - DBG_AUTH_SCRATCH [31:16] = DAR::AB[15:0]
5. On receiving EXIT_DBG_MB command, ROM exits the debug mailbox handler loop and continues normal boot flow.

51.7.6 41.9.6Debug Authentication Use cases

51.7.6.1 Return Material Analysis (RMA) Use case

The diagram shows RMA flow using debug authentication scheme, where a debug credential certificate is issued for each field technician.

1. Vendor generates RoT key pairs and programs the device with SHA256 hash of RoT public key hashes before shipping
2. Field technician generates his own key pair and provides public key to vendor for authorization.
3. Vendor attests the field technician’s public key. In the debug credential certificate, vendor assigns the access rights.
4. End customer having issues with a locked product takes it to field technician.
   - Field technician uses his credentials to authenticate with device and un-locks the product for debugging.

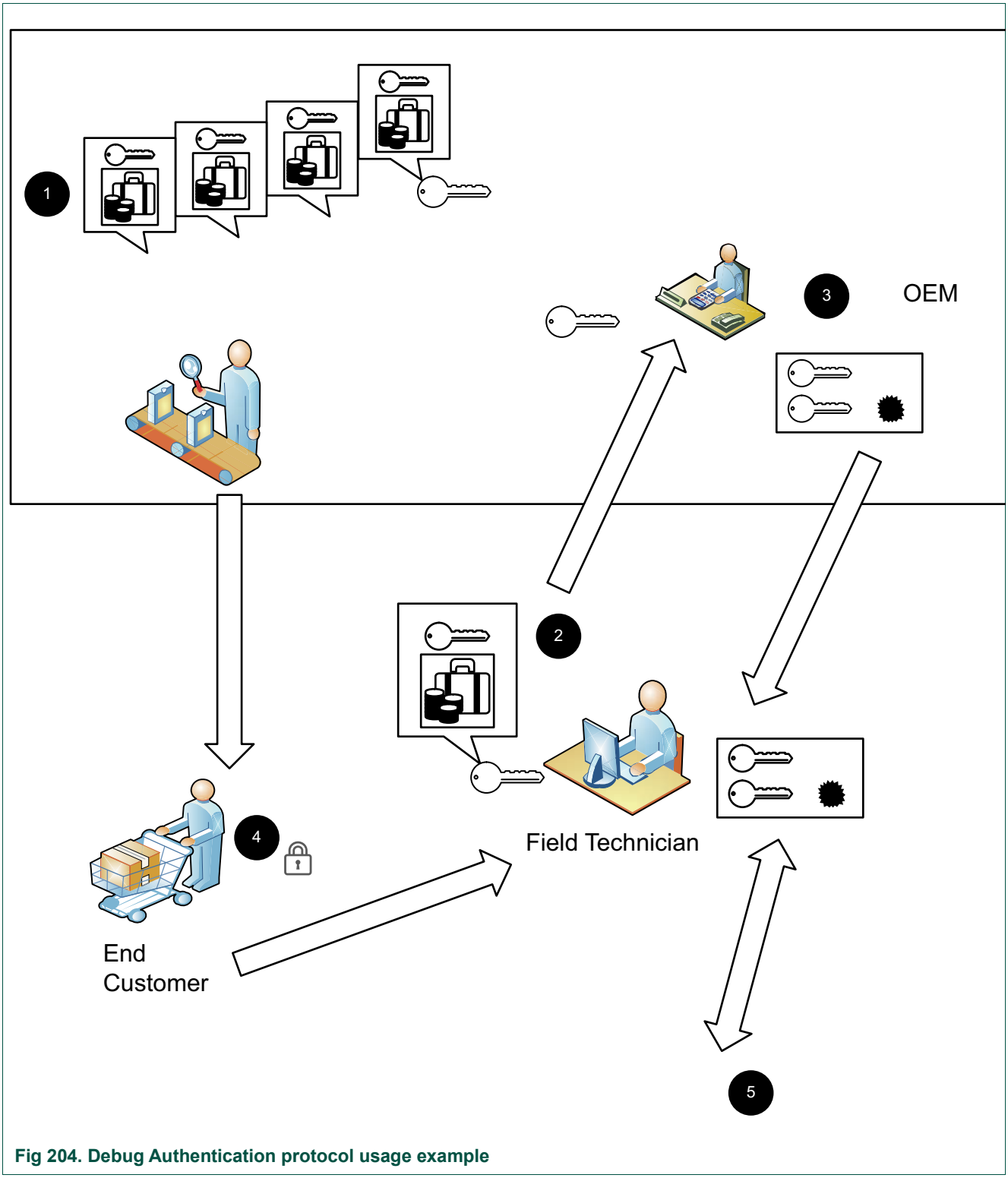


Fig 204. Debug Authentication protocol usage example

51.7.6.2 Module use case with OEM tier1 and teir2 Lifecycle states

The

CC_SOCU_PIN_NS & CC_SOCU_DFLT_NS

is provided to allow module-maker and OEM using the module to implement tiered protection approach.

- The module maker who is referred as Tier-1 developer can develop secure code and define access rights to his module using CC_SOCU_PIN & CC_SOCU_DFLT.
- Configuration can be such that debug access to secure module is blocked but non-secure debug is always allowed.
- Once the module is ready, Tier-1 developer can release the module to OEM (a.k.a. tier-2 developer), but block debug access to secure mode and enable debug access to non-secure mode.
- Tier-2 developer can develop non-secure module and extend access rights configuration to that module using CC_SOCU_DFLT_NS and CC_SOCU_PIN_NS.

51.7.7 41.9.7Glossary

Table 1088.41.9.7Glossary

Abbreviation	Term	Description
RoT	Root of Trust	Vendor-owned key pair that authorizes data assets via cryptographic signatures. The public part of the key is typically pre-configured in products so that data from untrusted sources can be cryptographically verified. The vendor public key used by the device to verify the signature of this DC. (The corresponding private key was used to sign the DC.)
RoTpub	RoT Public Key	The vendor public key used by the device to verify the signature of this DC. (The corresponding private key was used to sign the DC.)
RoTID	RoT Identifier	RoTID allows the debugger to infer which RoT public key(s) are acceptable to the device. If the debugger cannot or does not provide such a credential, the authentication process will fail.
RoTMETA	RoT meta-data	The RoT meta-data required by the device to corroborate; the ROTID sent in the DAC, the field in this DC, and any additional RoT state that is not stored within the device. This allows different RoT identification, management and revocation solutions to be handled.
SoCC	SoC Class	A unique identifier for a set of SoCs that require no SoC-specific differentiation in their debug authentication. The main usage is to allow a different set of debug domains and options to be negotiated between the device configuration and credentials. A class can contain just a single revision of a single SoC model, if the granularity of debug control warrants it.
DCK	Debug Credential Key	A user-owned key pair. The public part of the key is associated with a DC, the private part is held by the user and used to produce signatures during authentication.
DC	Debug Credential	A user public key and associated attributes, bound together and signed by a RoT, serves as an identity.
CC	Credential Constraint	In product configuration, CCs are limitations on the DCs that the device will accept for authentication. In DCs, CCs are vendor/RoT-authorized usages of the DC, as well as inputs to the desired debug behavior.
VU	Vendor Usage	A CC (constraint) value that is opaque to the debug authentication protocol itself but which can be leveraged by vendors in product-specific ways.
SoCU	SoC Usage	A CC (constraint) value that is a bit mask, and whose bits are used in an SoCC-specific manner. These bits are typically used for controlling which debug domains are accessed via the authentication protocol, but device-specific debug options can be managed in this way also.

Table 1088.41.9.7Glossary *…continued*

Abbreviation	Term	Description
CB AB	Credential Beacon Authentication beacon	A value that is passed through the authentication protocol, which is not interpreted by the protocol but is instead made visible to the application being debugged. A credential beacon is associated with a DC and is therefore vendor/RoT-signed. An authentication beacon is provided and signed by the debugger during the authentication process.
DCFG_*	Debug Config	Refers to device configuration settings stored in OTP.
CMPA		Customer Manufacturing Programmable area.
CFPA		Customer Field Programmable area.

52.1 How to read this chapter

The Inter-CPU Mailbox is available on LPC55S6x/LPC55S2x/LPC552x part.

52.2 Features

- Provides a means Inter-Processor Communication, allowing multiple CPUs to share resources and communicate with each other in a simple manner.
- Each CPU can cause up to thirty-two user defined interrupts to its partner.
- Each CPU can claim a shared resource if it is available.

52.3 Basic configuration

Set the MAILBOX bit in the AHBCLKCTRL0 register, see Section 4.5.17 “AHB clock control 0” to enable the clock to the Mailbox. Enable the interrupt in the NVIC, see Table 8.

52.4 Pin description

The Mailbox has no configurable pins.

52.5 General description

The Mailbox provides a means for simple communication between CPUs.

52.6 Register description

Table 1089.Register overview: Mailbox (base address 0x5008 B000)

Name	Access	Offset	Description	Reset value	Section
IRQ0	R/W	0x000	Interrupt request register for the Cortex-M33 (CPU1).	0x0	52.6.1
IRQ0SET	WO	0x004	Set bits in IRQ0.	-	52.6.2
IRQ0CLR	WO	0x008	Clear bits in IRQ0.	-	52.6.3
IRQ1	R/W	0x010	Interrupt request register for the Cortex-M33 (CPU0).	0x0	52.6.4
IRQ1SET	WO	0x014	Set bits in IRQ1.	-	52.6.5
IRQ1CLR	WO	0x018	Clear bits in IRQ1.	-	52.6.6
MUTEX	R/W	0x0F8	Mutual exclusion register[1].	0x1	52.6.7

[1] Reading and writing have specific side effects, see detailed register description.

52.6.1 Cortex-M33 (CPU1) interrupt register

The IRQ0 register allows CPU0 to send interrupt requests to CPU1. It is intended to allow communication between CPUs. For example, one CPU could be handling certain peripherals and signaling another CPU when data is available. Each bit can represent a different situation. The use of this feature is entirely up to the user.

Table 1090.CPU1 interrupt register (IRQ0, offset = 0x000)

Bit	Symbol	Description	Reset value
31:0	INTREQ	If any bit is set, an interrupt request is sent to the CPU1 interrupt controller.	0x0

52.6.2 Cortex-M33 (CPU1) interrupt set register

The IRQ0SET register is used to set bits in the IRQ0 register.

Table 1091.CPU1 interrupt set register (IRQ0SET, offset = 0x004)

Bit	Symbol	Description	Reset value
31:0	INTREQSET	Writing 1 sets the corresponding bit in the IRQ0 register.	-

52.6.3 Cortex-M33 (CPU1) interrupt clear register

The IRQ0CLR register is used to clear bits in the IRQ0 register.

Table 1092.CPU0 interrupt clear register (IRQ0CLR, offset = 0x008)

Bit	Symbol	Description	Reset value
31:0	INTREQCLR	Writing 1 clears the corresponding bit in the IRQ0 register.	-

52.6.4 Cortex-M33 (CPU0) interrupt register

The IRQ1 register allows CPU1 to send interrupt requests to CPU0. It is intended to allow communication between CPUs. For example, one CPU could be handling certain peripherals and signaling another CPU when data is available. Each bit can represent a different situation. The use of this feature is entirely up to the user.

Table 1093.CPU0 interrupt (IRQ1, offset = 0x010)

Bit	Symbol	Description	Reset value
31:0	INTREQ	If any bit is set, an interrupt request is sent to the Cortex-M33 (CPU0) interrupt controller.	0x0

52.6.5 Cortex-M33 (CPU0) interrupt set register

The IRQ1SET register is used to set bits in the IRQ1 register.

Table 1094.CPU0 interrupt set register (IRQ1SET, offset = 0x014)

Bit	Symbol	Description	Reset value
31:0	INTREQSET	Writing 1 sets the corresponding bit in the IRQ1 register.	-

52.6.6 Cortex-M33 (CPU0) interrupt clear register

The IRQ1CLR register is used to clear bits in the IRQ1 register.

Table 1095.CPU0 interrupt clear register (IRQ1CLR, offset = 0x018)

Bit	Symbol	Description	Reset value
31:0	INTREQCLR	Writing 1 clears the corresponding bit in the IRQ1 register.	-

52.6.7 Mutual exclusion register

This register provides an Inter-Processor Communication handshake. When read for any reason, the current value will be returned and the bit will be cleared. The bit will be set again following any write.

It can be used as a resource allocation handshake between two CPUs. Whenever a CPU wishes to access a shared resource (possibly a resource allocation table in memory), it reads the MUTEX register. If it encounters a 1, it has control over the shared resource allocation. When it has made any needed changes, it writes to the register, causing the EX bit to become set again, and making control of shared resource allocation available to another CPU. If a CPU reads a 0, it must wait for the bit to read as a 1 before accessing the shared resource allocation information.

Table 1096.Mutual exclusion register (MUTEX, offset = 0x0F8)

Bit	Symbol	Description	Reset value
0	EX	Cleared when read, set when written. See usage description above.	0x1
31:1	-	Reserved.	-

53.1 Abbreviations

Table 1097.Abbreviations

Acronym	Description
ADC	Analog-to-Digital Converter.
AHB	Advanced High-performance Bus.
AMBA	Advanced Micro-controller Bus Architecture.
APB	Advanced Peripheral Bus.
API	Application Programming Interface.
AVB	Audio Video Bridging.
BOD	BrownOut Detection.
Boot	At power-up or chip reset, any method of importing code from an external source to execute from on-chip SRAM, or code executed in place from the external memory.
BSDL	Boundary-Scan Description Language.
CRC	Cyclic Redundancy Check.
DCC	Debug Communication Channel.
DMA	Direct Memory Access.
EMC	External Memory Controller.
Ethernet AVB	Ethernet Audio Video Bridging .
FIFO	First-In-First-Out.
FRO oscillator	Internal Free-Running Oscillator, tuned to the factory specified frequency.
GPIO	General Purpose Input/Output.
I2C	Inter-IC Control bus.
I2C or IIC	Inter-Integrated Circuit bus.
IAP	In-Application Programming.
I2S	Inter-IC Sound or Integrated Interchip Sound. A serial audio data communication method.
IrDA	Infrared Data Association.
ISP	In-System Programming. These are methods of programming any on-chip memory on a device.
ISR	Interrupt Service Routine.
JTAG	Joint Test Action Group.
LIN	Local Interconnect Network.
NVIC	Nested Vectored Interrupt Controller.
PDM	Pulse Density Modulation. This is the data format used by the digital microphone inputs.
PLL	Phase-Locked Loop.
POR	Power-On Reset.
PWM	Pulse Width Modulator.
RAM	Random Access Memory.
SDIO	Secure Digital Input Output.
SPI	Serial Peripheral Interface.
SRAM	Static Random Access Memory.

Table 1097.Abbreviations *…continued*

Acronym	Description
SWD	Serial-Wire Debug.
TAP	Test Access Port.
USART	Universal Synchronous/Asynchronous Receiver/Transmitter.
VAD	Voice Activity Detect.

53.2 References

[1] **Cortex-M33 DGUG —** ARM Cortex-M33 Devices Generic User Guide

[2] **AN11538 —** AN11538 application note and code bundle (SCT cookbook)

[3] **UM10204 —** I^2C-bus specification and user manual

53.3 Tables

53.4 Figures

53.5 Contents

Chapter 1: LPC55S6x/LPC55S2x/LPC552x Introductory Information

Chapter 2: LPC55S6x/LPC55S2x/LPC552x Memory Map

Chapter 3: LPC55S6x/LPC55S2x/LPC552x Nested Vectored Interrupt Controller (NVIC)

Chapter 4: LPC55S6x/LPC55S2x/LPC552x SYSCON

continued >>

Chapter 5: LPC55S6x/LPC55S2x/LPC552x Flash

Chapter 6: LPC55S6x/LPC55S2x/LPC552x Boot ROM

Chapter 7: LPC55S6x/LPC55S2x/LPC552x Secure Boot ROM

Chapter 8: LPC55S6x/LPC55S2x/LPC552x ISP and IAP

Chapter 9: LPC55S6x/LPC55S2x/LPC552x Flash API

Chapter 10: LPC55S6x/LPC55S2x/LPC552x Protected Flash Region

Chapter 11: LPC55S6x/LPC55S2x/LPC552x Analog control

Chapter 12: LPC55S6x/LPC55S2x/LPC552x Cap Bank API

Chapter 13: LPC55S6x/LPC55S2x/LPC552x Power Management

Chapter 14: LPC55S6x/LPC55S2x/LPC552x Power Profiles/Power Control API

Chapter 15: LPC55S6x/LPC55S2x/LPC552x I/O Pin Configuration (IOCON)

Chapter 16: LPC55S6x/LPC55S2x/LPC552x General Purpose I/O (GPIO)

Chapter 17: LPC55S6x/LPC55S2x/LPC552x Secure General Purpose I/O (Secure GPIO)

Chapter 18: LPC55S6x/LPC55S2x/LPC552x Input Multiplexing (INPUTMUX)

Chapter 19: LPC55S6x/LPC55S2x/LPC552x Pin Interrupt and Pattern Match (PINT)

Chapter 20: LPC55S6x/LPC55S2x/LPC552x Secure pin interrupt and pattern match (Secure PINT)

Chapter 21: LPC55S6x/LPC55S2x/LPC552x:Group GPIO Input Interrupt (GINT0/1)

Chapter 22: LPC55S6x/LPC55S2x/LPC552x DMA controller

Chapter 23: LPC55S6x/LPC55S2x/LPC552x SDMMC and SDIO

Chapter 24: LPC55S6x/LPC55S2x/LPC552x SCTimer/PWM (SCT)

Chapter 25: LPC55S6x/LPC55S2x/LPC552x Standard counter/timers (CTIMER0 - 4)

Chapter 26: LPC55S6x/LPC55S2x/LPC552x Micro-tick Timer (UTICK)

Chapter 27: LPC55S6x/LPC55S2x/LPC552x Multi-Rate Timer (MRT)

Chapter 28: LPC55S6x/LPC55S2x/LPC552x Real-Time Clock (RTC)

Chapter 29: LPC55S6x/LPC55S2x/LPC552x System Tick Timer

Chapter 30: LPC55S6x/LPC55S2x/LPC552x Windowed Watchdog Timer (WWDT)

Chapter 31: LPC55S6x/LPC55S2x/LPC552x OS Event Timer

Chapter 32: LPC55S6x/LPC55S2x/LPC552x Flexcomm Interface Serial Communication

Chapter 33: LPC55S6x/LPC55S2x/LPC552x I²C-bus Interfaces

Chapter 34: LPC55S6x/LPC55S2x/LPC552x USARTs

Chapter 35: LPC55S6x/LPC55S2x/LPC552x Serial Peripheral Interfaces

Chapter 36: LPC55S6x/LPC55S2x/LPC552x Sys_ctrl

Chapter 37: LPC55S6x/LPC55S2x/LPC552x I²S interface

Chapter 38: LPC55S6x/LPC55S2x/LPC552x Programmable Logical Unit (PLU)

Chapter 39: LPC55S6x/LPC55S2x/LPC552x 16-bit ADC controller (ADC)

Chapter 40: LPC55S6x/LPC55S2x/LPC552x Analog Comparator

Chapter 41: LPC55S6x/LPC55S2x/LPC552x USB0 full-speed device controller

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 3 October 2024
Document identifier: UM11126

Chapter 42: LPC55S6x/LPC55S2x/LPC552x USB0 Full-Speed Host Controller

Chapter 43: LPC55S6x/LPC55S2x/LPC552x USB1 High-Speed Host Controller

Chapter 44: LPC55S6x/LPC55S2x/LPC552x USB1 High-Speed Device Controller

Chapter 45: LPC55S6x/LPC55S2x/LPC552x USB1 High-Speed PHY

Chapter 46: LPC55S6x/LPC55S2x/LPC552x CRC engine

Chapter 47: LPC55S6x/LPC55S2x/LPC552x Trusted Execution Environment

Chapter 48: LPC55S6x/LPC55S2x/LPC552x Security features

Chapter 49: LPC55S6x/LPC55S2x/LPC552x PowerQuad DSP Coprocessor and Accelerator

Chapter 50: LPC55S6x/LPC55S2x/LPC552x Cryptographic Accelerator CASPER

Chapter 51: LPC55S6x/LPC55S2x/LPC552x Debug Subsystem

Chapter 52: LPC55S6x/LPC55S2x/LPC552x Inter-CPU Mailbox

Chapter 53: Supplementary information

Legal information

Definitions

Draft — A draft status on a document indicates that 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 in a draft version of a document and shall have no liability for the consequences of use of such information.

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.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

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.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

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

Date of release: 3 October 2024
Document identifier: UM11126