UM11424

LPC55S0x/LPC550x User manual

Rev. 1.5 — 21 December 2023 User manual

Document information

Info	Content
Keywords	LPC55S06JBD64, LPC55S06JHI48, LPC55S04JBD64, LPC55S04JHI48, LPC5506JBD64, LPC5506JHI48, LPC5504JBD64, LPC5504JHI48, LPC5502JBD64, LPC5502JHI48, ARM Cortex-M33 core, 32-bit microcontroller, SCTimer/PWM, PLU, TrustZone, CASPER, I2C, AES, PUF, SHA, CRC, RNG, 16-bit ADC, CAN-FD
Abstract	LPC55S0x/LPC550x User Manual

Contact information

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

Revision history

Rev	Date	Description
1.5	20231221	• Added a note in Section 44.9 “DICE”. • Updated CMD[CTYPE] register definition in Table 728 “ADC command low buffer registers (CMDL[1:15], offsets 0x100 to 0x170)”.

Revision history *…continued*

Rev	Date	Description
1.4	20230419	• In Figure 18 “Reserved RAM region for the boot ROM” updated value to 0x30007FFF. • Removed System Device Id from Table 254 “Properties used by Get/SetProperty commands, sorted by values”. • Added 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” in Section 39.3 “Basic configuration” and Section 39.7.6 “Temperature sensor”. • Added Remark “For best ADC performance, force the offset calibration to -16. Set this prior to performing the gain calibration” in Section 39.3 “Basic configuration” and Section 39.7.6 “Temperature sensor”. • Updated RTC nReset from “Rtc” to “Act” in Table 164 “Resets”. • Updated “SPI_FLASH_CFG (0x9E404)” to “SPI_FLASH_CFG (0x3E404)” in Section 6.4.3 “SPI flash recovery”. • Replaced “the image can be programmed into internal flash or booted into internal SRAM” to “the image can be booted into internal SRAM in a non-reserved region” in Section 6.4.3 “SPI flash recovery”. • Updated commands for SB file recovery mode in Section 6.4.3 “SPI flash recovery”. • Added 19:16 bit description “When DMA operations are enabled in ADC registers (FWMDE0 = 1 in DE register and FWMDE =1 in DE register), the ADC FIFO watermark level must be set to a minimum value of 2” in Table 725 “ADC FIFO control registers (FCTRL[0:1], offsets 0xE0 to 0xE4)”. • Updated from “Two MCAN Controllers” to “One MCAN Controller” in Section 41.7 “Register description”. • Updated SOCU_DFLT [n] value for Restriction level 0 and 3 in Table 1000 “Access restriction levels”. • Updated “0x30000000 to 0x30003FFF” to “0x30000000 to 0x30007FFF” in Section 7.4.1 “skboot_authenticate”. • Replaced “RETURN_EN” to “ENABLE_FA_MODE” in Table 279 “Lifecycle state descriptions”. • Updated Section 4.5.46 “Fractional rate divider for each Flexcomm Interface frequency”. • Updated remark in Section 33.4 “Basic configuration”. • Updated Figure 95 “Select clock for FCLK” and Section 34.3 “Basic configuration”. • Updated remark in Section 35.3 “Basic configuration”. • Updated Section 37.3 “Basic configuration”. • Updated Remark in Section 5.3 “Block diagram”. • Added text “(default is PIO0_0)” in Section 10.2.1 “Customer Manufacturing Programmable Area (CMPA)”. • Updated Table 477 “Suggested SCT input pin settings” and Table 478 “Suggested SCT output pin settings” in Section 23.4 “Pin description”. • Updated Table 623 “Suggested USART pin settings” in Section 34.4 “Pin description”. • Updated Section 13.3.5 “Power-down mode”. • Added Table 300 “Power domain supply”. • Replace “FRO1M” to “Xtal clock coming directly” in Section 39.7.4 “Clock operation”. • Updated Table 344 “Parameter sram_retention_ctrl”,Table 337 “Parameter sram_retention_ctrl” and Table 346 “Parameter sram_retention_ctrl”. • Updated Section 4.5.62.2 “PLL0 status register” and Section 4.6.6.2.1 “Lock detector”. • Updated Table 647 “Suggested SPI pin settings” in Section 35.4 “Pin description”.

Revision history *…continued*

Rev	Date	Description
1.3	20211007	• Updated minimum sampling time to 3.5 ADCK cycles in Table 729 “ADC command high buffer registers (CMDH[1:15], offsets 0x104 to 0x174)”. • Updated Device_id0 register values in Table 162 “Device ID0 register (DEVICE_ID0, offset = 0xFF8) bit description”. • HS_SPI functions removed from pin P0_1, P0_2, and P0_3 in Table 358 “I/O control registers: FUNC values (FUNC = 5 to 9) and pin functions”. • Updated bit fields in Table 1003 “CC_LIST_Table”. • Updated LOCK REG values in Table 193 “PRINCE configuration”. • Updated Table 295 “Auxiliary bias register (AUX_BIAS, offset = 0xB4) bit description”. • CAN controller DBTP register range in DTSEG1 field updated to read 0 to 31 in Table 749 “Data bit timing and prescaler register (DBTP, offset 0x00C) bit description”. • Note extended in Section 14.4.7.2 “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 480 “SCT configuration register (CONFIG, offset = 0x000)”. • Figure 48 “SCT0 input multiplexing” of Chapter 18 “LPC55S0x/LPC550x Input Multiplexing (INPUTMUX)” updated from SCTASYNCCLK PLL to CGU SCTASYNCCLK. • FRG maximum allowed output frequency updated in Section 4.5.46 “Fractional rate divider for each Flexcomm Interface frequency”. • In Chapter 11 “LPC55S0x/LPC550x Analog control” updated Table 281 “(ANALOG_CTRL_CFG, offset = 0x0) bit description”, Table 286 “ADC control static configuration register (ADC_CTRL, offset = 0x18) bit description”, Table 288 “32 MHz Crystal oscillator status register (XO32M_STATUS, offset = 0x24) bit description”, Table 291 “First Ring Oscillator module control register (RINGO0_CTRL), offset = 0x40) bit description”, Table 292 “First Ring Oscillator module control register (RINGO1_CTRL), offset = 0x44) bit description”Table 293 “First Ring Oscillator module control register (RINGO2_CTRL), offset = 0x48) bit description”, and Table 294 “High Speed Crystal Oscillator Voltage Source Supply Control register (LDO_XO32M, offset = 0xB0) bit description”. • In Chapter 13 “LPC55S0x/LPC550x Power Management” updated Table 304 “Power Management Controller FSM (Finite State Machines) status (STATUS, offset = 0x4) bit description”, Table 306 “DCDC (first) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (DCDC0), offset = 0x10) bit description”, Table 307 “DCDC (second) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (DCDC1), offset = 0x14) bit description”, Table 310 “Analog References fast wake-up Control register (REFFASTWKUP, offset = 0x40) bit description”, Table 311 “32 KHz Crystal oscillator (XTAL) control register [Reset by: PoR, Brown Out Detectors Reset] (XTAL, offset = 0x4C) bit description”, Table 316 “Dummy Control bus to PMU [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (MISCCTRL, offset = 0x90) bit description”, and Table 322 “SRAM control register (SRAMCTRL, offset = 0xD4) bit description”.

Revision history *…continued*

Rev	Date	Description
1.2	20210520	Added DICE information under Table 39 “Register overview: SYSCON (base address = 0x50000000) bit description”, Section 4.5.65 “DICE register”, and Section 44.9 “DICE”. Made RNG updates to Chapter 44 “LPC55S0x Security features”.
1.1	20210429	Added footnotes to the end of Table 822 “Register overview: AHB_Secure_CTRL (base address = 0x400AC000)”, stating: 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.
1.0	20201020	Initial version.

UM11424

Chapter 1: LPC55S0x/LPC550x Introductory Information

Rev. 1.5 — 21 December 2023 **User manual**

1.1 Introduction

The LPC55S0x/LPC550x is an Arm Cortex®-M33 based micro-controller for embedded applications and includes the following features:

- Up to 96 kB of on-chip SRAM.
- Up to 256 kB on-chip flash.
- CAN-FD.
- Five general-purpose timers.
- SCTimer/PWM.
- RTC/alarm timer.
- 24-bit Multi-Rate Timer (MRT).
- Windowed Watchdog Timer (WWDT).
- Code Watchdog
- High speed SPI (50MHz).
- Eight flexible serial communication peripherals (each of which can be a USART, SPI, I2C, or I2s interface).
- 16-bit 2.0 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 LPC55S0x/LPC550x also offers support for HASH, AES, RSA, UUID, DICE, dynamic encrypt and decrypt, debug authentication, and TBSA compliance.

1.2 Features

- ARM Cortex-M33 core (CPU0, r0p4):
  – Running at a frequency of up to 96 MHz.
  – TrustZone, Floating Point Unit (FPU) and Memory Protection Unit (MPU).
  – ARM Cortex M33 built-in Nested Vectored Interrupt Controller (NVIC).
  – 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.
- CASPER crypto co-processor is provided to enable hardware acceleration for various functions required for asymmetric cryptographic algorithms, such as Elliptic Curve Cryptography (ECC).

- On-chip memory:
  - Up to 256 kB on-chip flash program memory with flash accelerator and 512 byte page erase and write.
  - Up to 96 kB total SRAM consisting of 16 kB SRAM on Code Bus, 64 kB SRAM on System Bus (64 kB is contiguous), and additional 16 kB SRAM on System Bus.
- PRINCE module for real-time encryption of data being written to on-chip flash and decryption of encrypted flash data during read operations 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 the following interfaces: UART interface (Flexcomm 0) with auto baud, SPI slave interfaces (Flexcomm 3 or 9) 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 the NXP Secure Boot file format, version 2.0 (SB2 files).
  - Supports booting of images from PRINCE encrypted flash regions.
  - Supports 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^{2}C, and I^{2}S interface. Each Flexcomm Interface includes a FIFO that supports USART, SPI, and I^{2}S. 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^{2}S.

  - 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.
- 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.
  - CRC engine block can calculate a CRC on supplied data using one of three standard polynomials with DMA support.
  - Up to 45 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.
  - AES-256 encryption/decryption engine with keys fed directly from PUF or a software supplied key.
  - Secure Hash Algorithm (SHA2) module supports secure boot with dedicated DMA controller.
  - 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.
  - True Random Number Generator (TRNG).
  - 128-bit unique device serial number for identification (UUID).
  - Secure GPIO.
  - Code Watchdog for detecting code flow integrity.

- 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 four 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 2.0 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 +/- 1% accuracy over the entire voltage and 0 C to 85 C. The FRO is trimmed to +/- 2% accuracy over the entire voltage and -40 C to 105 C.
  - 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.
  - High-speed 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.

  - 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 which can be used as an integrated load capacitor. Using APIs, the capacitor banks on each crystal pin can tune the frequency for crystals with a Capacitive Load (CL) which conserves board space and reduces 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 and internal CORE voltage with separate thresholds for forced reset.
- 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 HTQFP64 and HVQFN48 packages.

1.3 Block diagram

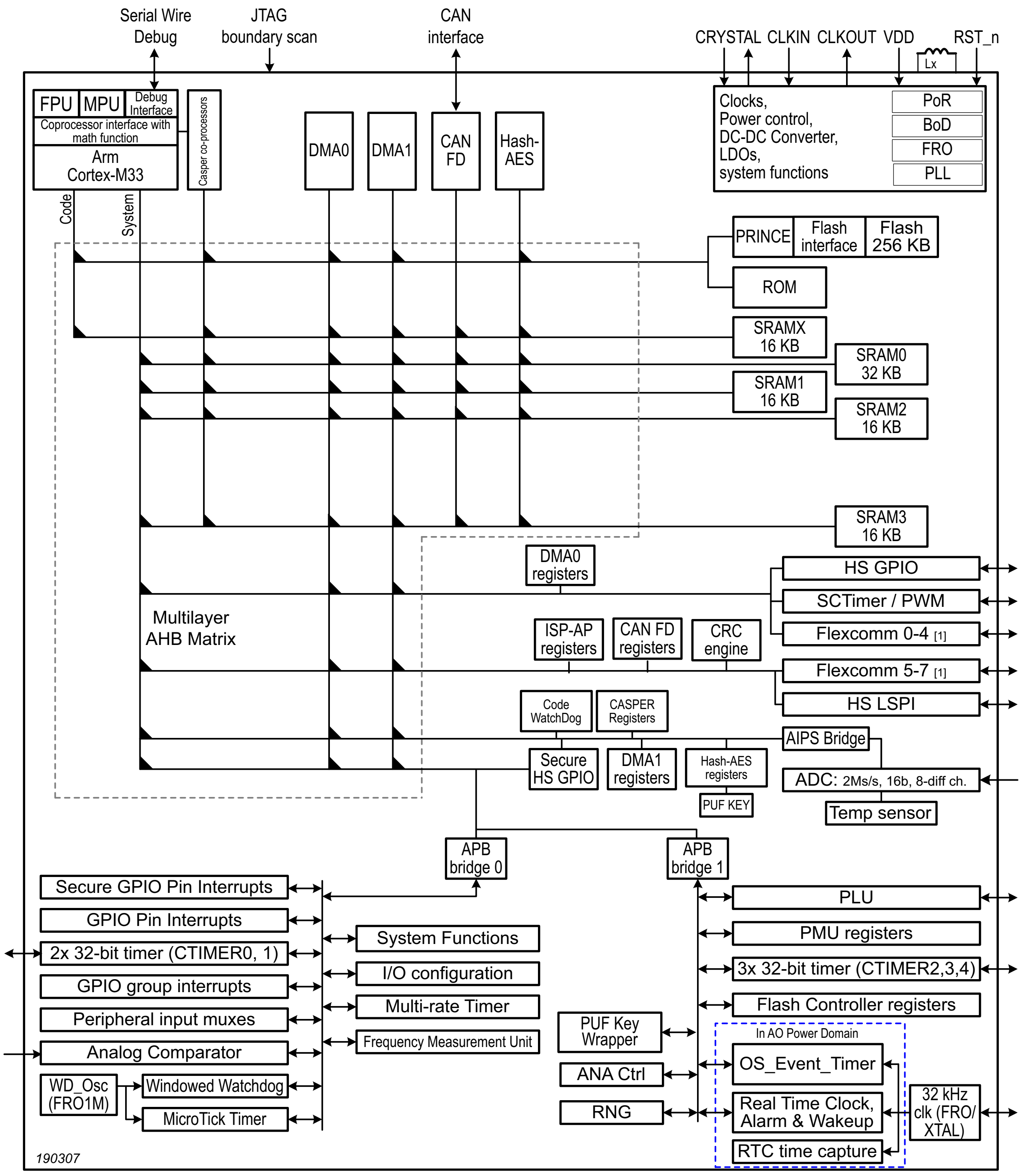


Fig 1. LPC55S0x/LPC550x block diagram

1.4 Architectural overview

The Arm Cortex M33 includes two AHB-Lite buses, one system bus and one code 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 LPC55S0x/LPC550x 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 On-chip Static RAM

The LPC55S0x/LPC550x supports 96 KB SRAM with separate bus master access for higher throughput and individual power control for low-power operation.

1.8 Ordering information

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
LPC55S06JBD64	HTQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 0.5mm pitch	SOT 855-5
LPC55S06JHI48	HVQFN48	plastic thermal enhanced very thin quad flat package; no leads; 48 terminals; body 7 x7 x 0.85 mm	SOT619-28
LPC55S04JBD64	HTQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 0.5mm pitch	SOT 855-5
LPC55S04JHI48	HVQFN48	plastic thermal enhanced very thin quad flat package; no leads; 48 terminals; body 7 x7 x 0.85 mm	SOT619-28
LPC5506JBD64	HTQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 0.5mm pitch	SOT 855-5
LPC5506JHI48	HVQFN48	plastic thermal enhanced very thin quad flat package; no leads; 48 terminals; body 7 x7 x 0.85 mm	SOT619-28
LPC5504JBD64	HTQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 0.5mm pitch	SOT 855-5
LPC5504JHI48	HVQFN48	plastic thermal enhanced very thin quad flat package; no leads; 48 terminals; body 7 x7 x 0.85 mm	SOT619-28
LPC5502JBD64	HTQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 0.5mm pitch	SOT 855-5
LPC5502JHI48	HVQFN48	plastic thermal enhanced very thin quad flat package; no leads; 48 terminals; body 7 x7 x 0.85 mm	SOT619-28

1.8.1 Ordering options

Table 2. Ordering options

Type number	Flash/KB	Total SRAM/KB	Secure boot	TrustZone	PUF Controller	HASH-AES	RNG	CASPER	PRINCE	CAN FD	GPIO	ADC Channels
LPC55S06JBD64	256	96	yes	yes	yes	yes	yes	yes	yes	CAN FD	45	9
LPC55S06JHI48	256	96	yes	yes	yes	yes	yes	yes	yes	CAN FD	30	7
LPC55S04JBD64	128	80	yes	yes	yes	yes	yes	yes	yes	CAN FD	45	9
LPC55S04JHI48	128	80	yes	yes	yes	yes	yes	yes	yes	CAN FD	30	7
LPC5506JBD64	256	96	-	-	-	-	yes	-	-	CAN2.0	45	9
LPC5506JHI48	256	96	-	-	-	-	yes	-	-	CAN2.0	30	7
LPC5504JBD64	128	80	-	-	-	-	yes	-	-	CAN2.0	45	9
LPC5504JHI48	128	80	-	-	-	-	yes	-	-	CAN2.0	30	7
LPC5502JBD64	64	48	-	-	-	-	yes	-	-	CAN2.0	45	9
LPC5502JHI48	64	48	-	-	-	-	yes	-	-	CAN2.0	30	7

Note:

- HTQFP64: up to 8 Flexcomm interfaces (UART up to 8, I2C up to 8, I2S up to 8 and SPI up to 6) + 1 HS SPI.
- HVQFN48: up to 7 Flexcomm interfaces (UART up to 7, I2C up to 7, I2S up to 4 and SPI up to 3) + 1 HS SPI.

UM11424

Chapter 2: LPC55S0x/LPC550x Memory Map

Rev. 1.5 — 21 December 2023 **User manual**

2.1 General description

2.1.1 AHB multilayer matrix

The LPC55S0x/LPC550x 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: 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	CPU bus	CM-33 usage
0x0000 0000	0x0FFF FFFF	Non-secure	Code	Flash memory, Boot ROM, SRAM X.
0x1000 0000	0x1FFF FFFF	Secure	Code	Flash memory, Boot ROM, SRAM X.
0x2000 0000	0x2FFF FFFF	Non-secure	Data	SRAM 0, SRAM 1, SRAM 2, SRAM 3.
0x3000 0000	0x3FFF FFFF	Secure	Data	SRAM 0, SRAM 1, SRAM 2, SRAM 3.
0x4000 0000	0x4FFF FFFF	Non-secure	Data	AHB and APB peripherals.
0x5000 0000	0x5FFF FFFF	Secure	Data	AHB and APB peripherals.

[1] The size shown for peripheral 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 provides a more detailed memory map as seen by the 2 Cortex-M33. 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	0x0003 FFFF	0x1000 0000	0x1003 FFFF	Flash memory, on CM33 code bus. The last 17 pages (12KB) are reserved on the 256 KB flash devices resulting in 244 KB internal flash memory.
	0x0300 0000	0x0301 FFFF	0x1300 0000	0x1301 FFFF	Boot ROM, on CM33 code bus.
1	0x0400 0000	0x0400 3FFF	0x1400 0000	0x1400 3FFF	SRAM X on CM33 code bus, 32 KB. SRAMX_0 (0x1400 0000 to 0x1400 0FFF) and SRAMX_1 (0x1400 1000 to 0x1400 1FFF) are used for Casper (total 8 KB). If CPU retention used in power-down mode, SRAMX_2 (0x1400 2000 to 0x1400 25FF) 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 7FFF	0x3000 0000	0x3000 7FFF	SRAM 0 on CM33 data bus, 32 KB.
3	0x2000 8000	0x2000 BFFF	0x3000 8000	0x3000 BFFF	SRAM 1 on CM33 data bus, 16 KB.
4	0x2000 C000	0x2000 FFFF	0x3000 C000	0x3000 FFFF	SRAM 2 on CM33 data bus, 16 KB.
5	0x2001 0000	0x2001 3FFF	0x3001 0000	0x3001 3FFF	SRAM 3, 16 KB.
6	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.
7	0x4008 0000	0x4008 FFFF	0x5008 0000	0x5008 FFFF	AHB peripherals. See Section 2.1.7.
8	0x4009 0000	0x4009 FFFF	0x5009 0000	0x5009 FFFF	AHB peripherals. See Section 2.1.7.
9	0x400A 0000	0x400A FFFF	0x500A 0000	0x500A 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.

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	Pin function selection and pin control setup (IOCON).
	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 (INPUTMUX).
	0x4000 7000	0x5000 7000	Reserved.
	0x4000 8000	0x5000 8000	Standard counter/timer 0 (CTimer0).
	0x4000 9000	0x5000 9000	Standard counter/timer 1 (CTimer1).
	0x4000 C000	0x5000 C000	Windowed watchdog timer 0 (WWDT0).
	0x4000 D000	0x5000 D000	Multi-Rate timer (MRT).
	0x4000 E000	0x5000 E000	Micro-Tick timer (Utick).
	0x4001 3000	0x5001 3000	Analog controls.
	0x4001 5000	0x5001 5000	Reserved.
1	0x4002 3000	0x5002 3000	I^{2}S signal sharing (Sysctl).
	0x4002 8000	0x5002 8000	Standard counter/timer 2 (Timer2).
	0x4002 9000	0x5002 9000	Standard counter/timer 3 (Timer3).
	0x4002 A000	0x5002 A000	Standard counter/timer 4 (Timer4).
	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	Reserved.
	0x4003 A000	0x5003 A000	True Random Number Generator.
	0x4003 B000	0x5003 B000	Physical Unclonable Function (PUF).
	0x4003 D000	0x5003 D000	Programmable Logic Unit (PLU).

2.1.7 AHB peripherals

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

Table 6. AHB peripheral memory map

AHB port	Non-secure base address	Secure base address	Peripheral
7	0x4008 2000	0x5008 2000	DMA0 registers.
	0x4008 4000	0x5008 4000	Reserved.
	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 C000	0x5008 C000	High-Speed GPIO.
8	0x4009 4000	0x5009 4000	Reserved.
	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 D000	0x5009 C000	Debug Mailbox (DM-AP).
	0x4009 C000	0x5009 D000	CAN0.
	0x4009 F000	0x5009 F000	High Speed SPI (SPI8).
9	0x400A 0000	0x500A 0000	ADC0.
	0x400A 1000	0x500A 1000	Code Watchdog (CDOG).
	0x400A 2000	0x500A 2000	Reserved.
	0x400A 3000	0x500A 3000	Reserved.
	0x400A 4000	0x500A 4000	Hash-AES registers.
	0x400A 5000	0x500A 5000	Casper.
	0x400A 7000	0x500A 7000	DMA1 registers.
	0x400A 8000	0x500A 8000	Secure HS GPIO.
	0x400A C000	0x500A C000	Security control registers.

2.1.8 RAM configuration

Table 7 describes the RAM configuration for the LPC55S0x/LPC550x.

Table 7. RAM Configuration

RAM Total	RAM-X (KB)	RAM0 (KB)	RAM1 (KB)	RAM2 (KB)	RAM3 (KB)
96 KB devices	16	32	16	16	16
80 KB devices	16	32	16	16 (for 80k; it is RAMx+RAM0+RAM1+RAM2)	- (for 80k; it is RAMx+RAM0+RAM1+RAM2)
48 KB devices	16	32	-	-	-

UM11424

Chapter 3: LPC55S0x/LPC550x Nested Vectored Interrupt Controller (NVIC)

Rev. 1.5 — 21 December 2023 **User manual**

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 the CPU.
- A tightly coupled interrupt controller provides low interrupt latency.
- Controls system exceptions and peripheral interrupts.
- The NVIC for 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 "GPIO grouped interrupt port enable registers".

3.3 General description

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

3.3.1 Interrupt sources

Table 8 lists the interrupt sources for each peripheral function. Each peripheral device can 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. As an example, in the case where two interrupts are given the same priority, the interrupt numbers shown in the table are relevant.

Table 8. Connection of interrupt sources to the NVIC

Interrupt	Name	Interrupt description	Flags
0	WDT_BOD_IRQn	Device specific interrupts.	Windowed watchdog timer, Brownout detect, Flash interrupt.
1	DMA0_IRQn	DMA0 controller.	Interrupt A and interrupt B, error interrupt.
2	GINT0_IRQn	GPIO group 0.	Enabled pin interrupts.
3	GINT1_IRQn	GPIO group 1.	Enabled pin interrupts.
4	PIN_INT0_IRQn	Pin interrupt 0 or pattern match engine slice 0.	PSTAT - pin interrupt status.

Table 8. Connection of interrupt sources to the NVIC

Interrupt	Name	Interrupt description	Flags
5	PIN_INT1_IRQn	Pin interrupt 1 or pattern match engine slice 1.	PSTAT - pin interrupt status.
6	PIN_INT2_IRQn	Pin interrupt 2 or pattern match engine slice 2.	PSTAT - pin interrupt status.
7	PIN_INT3_IRQn	Pin interrupt 3 or pattern match engine slice 3.	PSTAT - pin interrupt status.
8	UTICK0_IRQn	Micro-tick timer.	INTR.
9	MRT0_IRQn	Multi-rate timer.	Global MRT interrupts: GFLAG0, 1, 2, 3.
10	CTIMER0_IRQn	Standard counter/timer CTIMER0.	Match and capture interrupts.
11	CTIMER1_IRQn	Standard counter/timer CTIMER1.	Match and capture interrupts.
12	SCT0_IRQn	SCTimer/PWM.	EVFLAG SCT event.
13	CTIMER3_IRQn	Standard counter/timer CTIMER3	Match and capture interrupts.
14	FLEXCOMM0_IRQn	Flexcomm Interface 0 (USART, SPI, I^{2}C, I^{2}S, FLEXCOMM).	See enable read and set register of this module.
15	FLEXCOMM1_IRQn	Flexcomm Interface 1 (USART, SPI, I^{2}C, I^{2}S, FLEXCOMM).	Same as Flexcomm0.
16	FLEXCOMM2_IRQn	Flexcomm Interface 2 (USART, SPI, I^{2}C, I^{2}S, FLEXCOMM).	Same as Flexcomm0.
17	FLEXCOMM3_IRQn	Flexcomm Interface 3 (USART, SPI, I^{2}C, I^{2}S, FLEXCOMM).	Same as Flexcomm0.
18	FLEXCOMM4_IRQn	Flexcomm Interface 4 (USART, SPI, I^{2}C, I^{2}S, FLEXCOMM).	Same as Flexcomm0.
19	FLEXCOMM5_IRQn	Flexcomm Interface 5 (USART, SPI,I^{2}C, I^{2}S, FLEXCOMM).	Same as Flexcomm0.
20	FLEXCOMM6_IRQn	Flexcomm Interface 6 (USART, SPI, I^{2}C, I^{2}S, FLEXCOMM).	Same as Flexcomm0.
21	FLEXCOMM7_IRQn	Flexcomm Interface 7 (USART, SPI, I^{2}C, I^{2}S, FLEXCOMM).	Same as Flexcomm0.
22	ADC0_IRQn	ADC0.	See enable read and set register of this module.
23	Reserved39_IRQn	Reserved interrupt.	-
24	ACMP_IRQn	ACMP interrupts.	See enable read and set register of this module.
25	Reserved41_IRQn	Reserved interrupt.	-
26	Reserved42_IRQn	Reserved interrupt.	-
27	Reserved43_IRQn	Reserved interrupt.	-
28	Reserved44_IRQn	Reserved interrupt.	-
29	RTC_IRQn	RTC alarm and wake-up interrupts.	See enable read and set register of this module.
30	Reserved46_IRQn	Reserved interrupt.	-
31	WAKEUP_IRQn	Wakeup for low power mode (Power Down) use when wakeupio is enabled during power down.	See WAKEIOCAUSE register in power management chapter.
32	PIN_INT4_IRQn	Pin interrupt 4 or pattern match engine slice 4 int.	PSTAT - pin interrupt status.
33	PIN_INT5_IRQn	Pin interrupt 5 or pattern match engine slice 5 int.	PSTAT - pin interrupt status.
34	PIN_INT6_IRQn	Pin interrupt 6 or pattern match engine slice 6 int.	PSTAT - pin interrupt status.
35	PIN_INT7_IRQn	Pin interrupt 7 or pattern match engine slice 7 int.	PSTAT - pin interrupt status.

Table 8. Connection of interrupt sources to the NVIC

Interrupt	Name	Interrupt description	Flags
36	CTIMER2_IRQn	Standard counter/timer CTIMER2.	Match and capture interrupts.
37	CTIMER4_IRQn	Standard counter/timer CTIMER4.	Match and capture interrupts.
38	OS_EVENT_IRQn	OS_EVENT_TIMER and OS_EVENT_WAKEUP interrupts.	-
39	Reserved55_IRQn	Reserved interrupt.	-
40	Reserved56_IRQn	Reserved interrupt.	-
41	Reserved57_IRQn	Reserved interrupt.	-
42	Reserved58_IRQn	Reserved interrupt.	-
43	CAN0_IRQ0_IRQn	CAN Interrupt 0.	CAN Interrupts.
44	CAN0_IRQ1_IRQn	CAN Interrupt 1.	CAN Interrupts.
45	Reserved.	Reserved.	-
46	Reserved.	Reserved.	-
47	Reserved.	Reserved.	-
48	Reserved.	Reserved.	-
49	SEC_HYPERVISOR_CALL_IRQn	SEC_HYPERVISOR_CALL interrupt	HF interrupts.
50	SEC_GPIO_INT0_IRQ0_IRQn	Secure GPIO function is available on P0(0-31) and 2x Pin Interrupt outputs are available to NVIC.	SGPIO 0 interrupts.
51	SEC_GPIO_INT1_IRQ0_IRQn	Secure GPIO function is available on P0(0-31) and 2x Pin Interrupt outputs are available to NVIC.	SGPIO 1 interrupts.
52	PLU_IRQn	Programmable Logic Unit.	PLU interrupts.
53	SEC_VIO_IRQn	Secure violation interrupt.	Secure violation interrupts.
54	HASHCRYPT_IRQn	SHA interrupt.	Hash interrupts.
55	CASPER	CASPER Crypto co-processor interrupt.	Casper interrupts.
56	PUF	PUF Controller Interrupt.	PUF interrupts.
57	Reserved	-	-
58	SDMA1	Secure DMA (DMA1) controller.	Secure DMA interrupts.
59	HS_SPI	HS_SPI.	HS_SPI.
60	CDOG	CDOG Interrupt.	CDOG Interrupts.

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	0x100	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	0x104	Interrupt set enable register 1. See ISER0 description.	0	3.4.2
ICER0	R/W	0x180	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	0x184	Interrupt clear enable register 1. See ISER0 description.	0	3.4.4
ISPR0	R/W	0x200	Interrupt set pending register 0. This register allows changing the interrupt state to pending and reading back the interrupt pending state for peripheral functions.	0	3.4.5
ISPR1	R/W	0x204	Interrupt set pending register 1. See ISPR0 description.	0	3.4.6
ICPR0	R/W	0x280	Interrupt clear pending register 0. This register allows changing the interrupt state to not pending and reading back the interrupt pending state for peripheral functions.	0	3.4.7
ICPR1	R/W	0x284	Interrupt clear pending register 1. See ICPR0 description.	0	3.4.8
IABR0	RO	0x300	Interrupt active bit register 0. This register allows reading the current interrupt active state for specific peripheral functions.	0	3.4.9
IABR1	RO	0x304	Interrupt active bit register 1. See IABR0 description.	0	3.4.10
IPR0	R/W	0x400	Interrupt priority register 0. This register contains the 3-bit priority fields for interrupts 0 to 3.	0	3.4.11
IPR1	R/W	0x404	Interrupt priority register 1. This register contains the 3-bit priority fields for interrupts 4 to 7.	0	3.4.12
IPR2	R/W	0x408	Interrupt priority register2. This register contains the 3-bit priority fields for interrupts 8 to 11.	0	3.4.13
IPR3	R/W	0x40C	Interrupt priority register 3. This register contains the 3-bit priority fields for interrupts 12 to 15.	0	3.4.14
IPR4	R/W	0x410	Interrupt priority register 4. This register contains the 3-bit priority fields for interrupts 16 to 19.	0	3.4.15
IPR5	R/W	0x414	Interrupt priority register 5. This register contains the 3-bit priority fields for interrupts 20 to 23.	0	3.4.16
IPR6	R/W	0x418	Interrupt priority register 6. This register contains the 3-bit priority fields for interrupts 24 to 27.	0	3.4.17
IPR7	R/W	0x41C	Interrupt priority register 7. This register contains the 3-bit priority fields for interrupts 28 to 31.	0	3.4.18
IPR8	R/W	0x420	Interrupt priority register 8. This register contains the 3-bit priority fields for interrupts 32 to 35.	0	3.4.19
IPR9	R/W	0x424	Interrupt priority register 9. This register contains the 3-bit priority fields for interrupts 36 to 39.	0	3.4.20
IPR10	R/W	0x428	Interrupt priority register 10. This register contains the 3-bit priority fields for interrupts 40 to 43.	0	3.4.21

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

Name	Access	Offset	Description	Reset value	Section
IPR11	R/W	0x42C	Interrupt priority register 11. This register contains the 3-bit priority fields for interrupts 44 to 47.	0	3.4.22
IPR12	R/W	0x430	Interrupt priority register 12. This register contains the 3-bit priority fields for interrupts 48 to 51	0	3.4.23
IPR13	R/W	0x434	Interrupt priority register13. This register contains the 3-bit priority fields for interrupts 52 to 55.	0	3.4.24
IPR14	R/W	0x438	Interrupt priority register14. This register contains the 3-bit priority fields for interrupts 56 to 60.	0	3.4.25
IPR15	R/W	0x43C	Interrupt priority register15. This register contains the 3-bit priority fields for interrupts 61 to 63.	0	3.4.26
STIR	WO	0xF00	Software trigger interrupt register, allows software to generate interrupts.	-	3.4.27

3.4.1 Interrupt set-enable register 0

The ISER0 register enables the first 32 peripheral interrupts or provides the ability to read the enabled state of these interrupts. The remaining interrupts are enabled via the ISER1 register, see Section 3.4.2 “Interrupt set-enable register 1”. Interrupts are disabled 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 matrix secure violation.
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.

Table 10. Interrupt set-enable register 0

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

[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 is used to enable the second group of peripheral interrupts or to read the enabled state of these 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	-	[1]	-
11	ISE_CAN0_INT0	[1]	CAN0 interrupt 0.
12	ISE_CAN0_INT1	[1]	CAN0 interrupt 1.
13	-	[1]	Reserved.
14	-	[1]	Reserved.
15	-	[1]	Reserved.
16	-	[1]	Reserved.
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.

Table 11. Interrupt set-enable register 1

Bit	Name	Value	Function
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.
24	ISE_PUF	[1]	PUF interrupt enable.
25	-	[1]	-
26	ISE_ SDMA1	[1]	Secure DMA (DMA1) interrupt enable.
27	ISE_HS_SPI	[1]	FC8 interrupt enable.
60	ISE_CDOG	[1]	CDOG interrupt.

[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 is used to disable the first 32 peripheral interrupts or to read the enabled state of the interrupts. The remaining interrupts are disabled via the ICER1 register Section 3.4.4 “Interrupt clear enable register 1”. Interrupts are enabled 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 is used to disable the second group of peripheral interrupts or to read the enabled state of the interrupts. 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 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 clears the pending state of the first 32 peripheral interrupts or reads 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.

Table 22. Interrupt priority register 2

Bit	Name	Function
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.
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.
31:24	-	Unused.

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
7:0	-	Unused.
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	-	Unused.

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.
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	-	Unused.
28:24	-	Unused.
31:29	IP_CAN0_INT0	CAN0 INT0 interrupt priority. 0 = highest priority. 7 = lowest priority.

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	IP_CAN0_INT1	CAN0 INT1 interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused.
15:13	Reserved	-
31:16	-	Unused.

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
7:0	-	Unused.
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
7:0	-	Unused.
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	-	Unused.
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 Interrupt priority register 15

The IPR15 register controls the priority of the sixteenth group of four peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 35. Interrupt priority register 15

Bit	Name	Function
4:0	-	Unused.
7:5	IP_CDOG	CDOG interrupt priority. 0 = highest priority 1 = lowest priority.
12:8	-	Unused.
15:13	-	Unused.
20:16	-	Unused.
23:21	-	Unused.
28:24	-	Unused.
31:29	-	Unused.

3.4.27 Software trigger interrupt register

In addition to using the ISPR registers to generate an interrupt, the STIR register provides an alternate method for generating a software interrupt. 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 36.

Table 36. Software trigger interrupt register (STIR)

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

UM11424

Chapter 4: LPC55S0x/LPC550x SYSCON

Rev. 1.5 — 21 December 2023 User manual

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 the 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 “LPC55S0x/LPC550x 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”. $\overline{\text{RESET}}$ 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.

See Section 4.5.32 “Main clock source select register A” and Section 4.5.33 “Main clock source select register B”.

2. Select the divider value for the system clock Section 4.5.47 “AHB clock divider register”.
3. Enable the clock for 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 the 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” and Section 18.6.10 “Frequency measure function target clock select register”.

Details on the accuracy and measurement process are described in Section 4.6.5 “Flash accelerator functional description”.

4.3 Pin description

Table 37. SYSCON pin description

Function	Type	Pin	Description	Reference
CLKOUT	O	PIO0_16, PIO0_26	CLKOUT clock output.	Chapter 15 “LPC55S0x/LPC550x 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 2 gives an overview of potential clock options. Table 38 describes signals on the clocking diagram. The maximum clock frequency is 96 MHz.

Remark: The indicated clock multiplexers shown in Figure 2 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 the main clock is 150 MHz and system clock (to CPU0, AHB bus, Sync, etc.,) is 96 MHz. See Figure 2 “Clock generation (Part 1 of 2)”. The POWER_SetVoltageForFreq API call must always be used when setting or switching the frequency. See Chapter 14 “LPC55S0x/LPC550x Power Profiles/Power Control API”.

Table 38. 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 2.
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).
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 2.
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 2.
pll1_clk	The output of the PLL1. The PLL1 and its source selection is shown in Figure 2.
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.

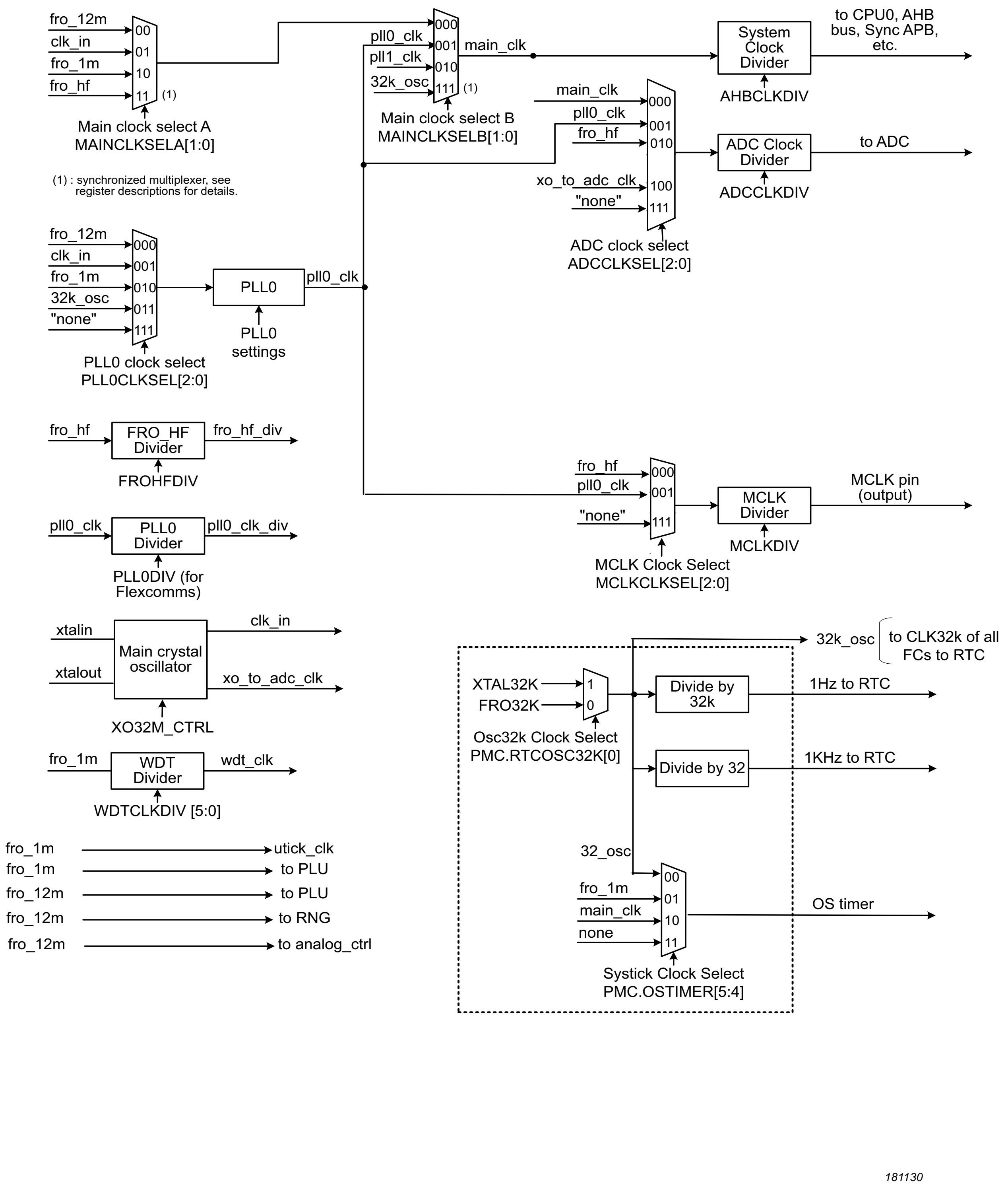


Fig 2. Clock generation (Part 1 of 2)

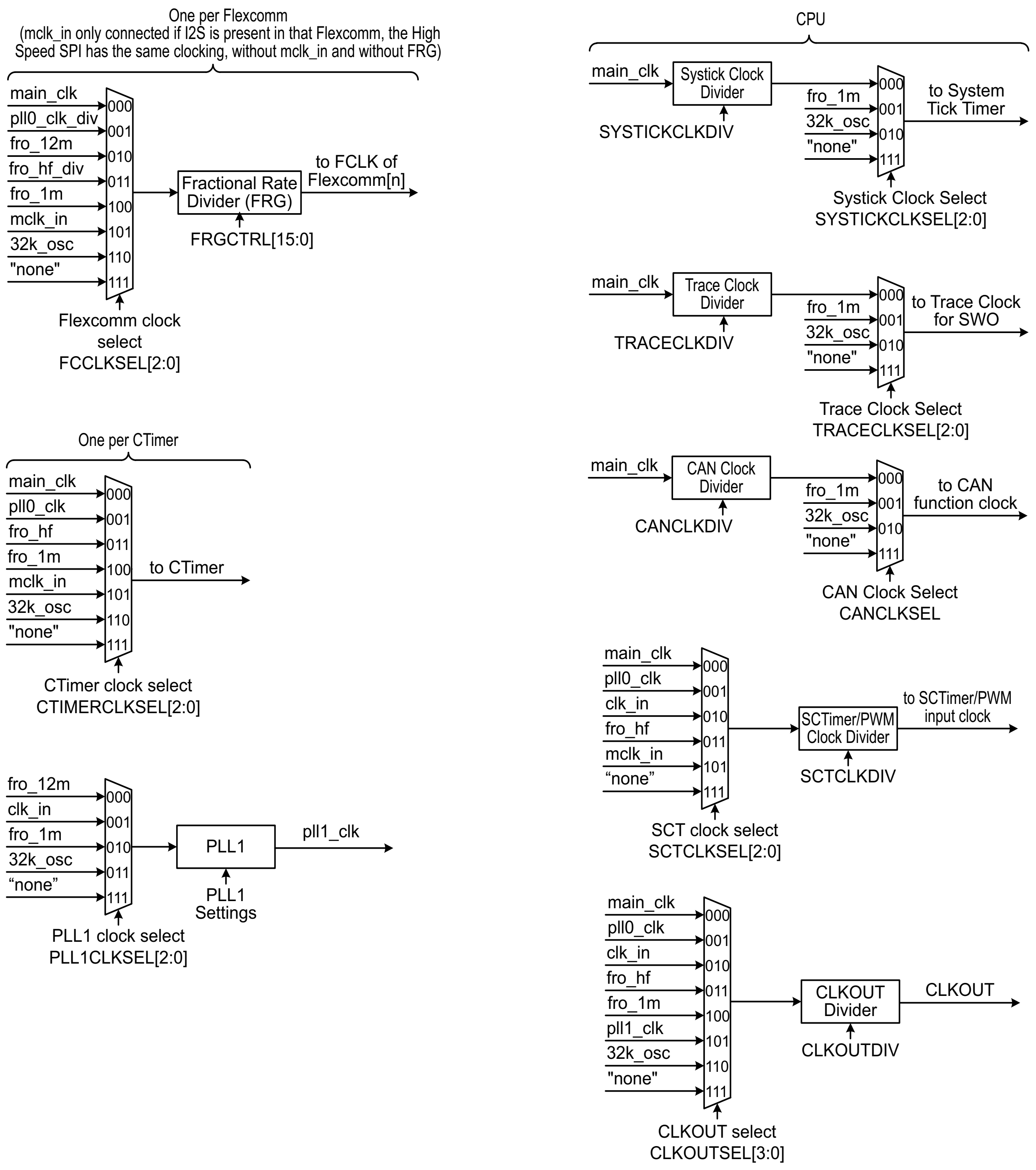


Fig 3. Clock generation (Part 2 of 2)

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 39 is secure and 0x4000 0000 is non-secure.

Note: 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 39. Register overview: SYSCON (base address = 0x50000000) bit description

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
NMISRC	RW	0x48	NMI source select.	0x0	4.5.5
PRESETCTRL0	RW	0x100	Peripheral reset control 0.	0x0	4.5.6
PRESETCTRL1	RW	0x104	Peripheral reset control 1.	0x0	4.5.7
PRESETCTRL2	RW	0x108	Peripheral reset control 2.	0x0	4.5.8
PRESETCTRLSET0	RW	0x120	Peripheral reset control set register.	0x0	4.5.9
PRESETCTRLSET1	RW	0x124	Peripheral reset control set register.	0x0	4.5.10
PRESETCTRLSET2	RW	0x128	Peripheral reset control set register.	0x0	4.5.11
PRESETCTRLCLR0	RW	0x140	Peripheral reset control clear register.	0x0	4.5.12
PRESETCTRLCLR1	RW	0x144	Peripheral reset control clear register.	0x0	4.5.13
PRESETCTRLCLR2	RW	0x148	Peripheral reset control clear register.	0x0	4.5.14
SWR_RESET	W	0x160	Generate a software reset.	0x0	4.5.15
AHBCLKCTRL0	RW	0x200	AHB clock control 0.	0x180	4.5.16
AHBCLKCTRL1	RW	0x204	AHB clock control 1.	0x0	4.5.17
AHBCLKCTRL2	RW	0x208	AHB clock control 2.	0x0	4.5.18
AHBCLKCTRLSET0	RW	0x220	Peripheral reset control register.	0x0	4.5.19
AHBCLKCTRLSET1	RW	0x224	Peripheral reset control register.	0x0	4.5.20
AHBCLKCTRLSET2	RW	0x228	Peripheral reset control register.	0x0	4.5.21
AHBCLKCTRLCLR0	RW	0x240	Peripheral reset control register.	0x0	4.5.22
AHBCLKCTRLCLR1	RW	0x244	Peripheral reset control register.	0x0	4.5.23
AHBCLKCTRLCLR2	RW	0x248	Peripheral reset control register.	0x0	4.5.24
SYSTICKCLKSEL0	RW	0x260	System Tick Timer for CPU0 source select.	0x0	4.5.25

Table 39. Register overview: SYSCON (base address = 0x50000000) bit description *…continued*

Name	Access	Offset	Description	Reset value	Section
TRACECLKSEL	RW	0x268	Trace clock source select.	0x0	4.5.26
CTIMERCLKSEL0	RW	0x26C	CTimer 0 clock source select.	0x0	4.5.27
CTIMERCLKSEL1	RW	0x270	CTimer 1 clock source select.	0x0	4.5.28
CTIMERCLKSEL2	RW	0x274	CTimer 2 clock source select.	0x0	4.5.29
CTIMERCLKSEL3	RW	0x278	CTimer 3 clock source select.	0x0	4.5.30
CTIMERCLKSEL4	RW	0x27C	CTimer 4 clock source select.	0x0	4.5.31
MAINCLKSELA	RW	0x280	Main clock A source select.	0x0	4.5.32
MAINCLKSELB	RW	0x284	Main clock B source select.	0x0	4.5.33
CLKOUTSEL	RW	0x288	CLKOUT clock source select.	0x7	4.5.34
PLL0CLKSEL	RW	0x290	PLL0 clock source select.	0x7	4.5.35
PLL1CLKSEL	RW	0x294	PLL1 clock source select.	0x7	4.5.36
CANCLKSEL	RW	0x2A0	CAN clock select.	0x7	4.5.37
ADCCLKSEL	RW	0x2A4	ADC clock source select.	0x7	4.5.38
FCCLKSEL0	RW	0x2B0	Flexcomm Interface 0 clock source select for Fractional Rate Divider.	0x7	4.5.39
FCCLKSEL1	RW	0x2B4	Flexcomm Interface 1 clock source select for Fractional Rate Divider.	0x7	4.5.39
FCCLKSEL2	RW	0x2B8	Flexcomm Interface 2 clock source select for Fractional Rate Divider.	0x7	4.5.39
FCCLKSEL3	RW	0x2BC	Flexcomm Interface 3 clock source select for Fractional Rate Divider.	0x7	4.5.39
FCCLKSEL4	RW	0x2C0	Flexcomm Interface 4 clock source select for Fractional Rate Divider.	0x7	4.5.39
FCCLKSEL5	RW	0x2C4	Flexcomm Interface 5 clock source select for Fractional Rate Divider.	0x7	4.5.39
FCCLKSEL6	RW	0x2C8	Flexcomm Interface 6 clock source select for Fractional Rate Divider.	0x7	4.5.39
FCCLKSEL7	RW	0x2CC	Flexcomm Interface 7 clock source select for Fractional Rate Divider.	0x7	4.5.39
HSLSPICLKSEL	RW	0x2D0	HS SPI clock source select.	0x7	4.5.40
MCLKCLKSEL	RW	0x2E0	I^{2}S MCLK clock source select.	0x7	4.5.41
SCTCLKSEL	RW	0x2F0	SCTimer/PWM clock source select.	0x7	4.5.42
SYSTICKCLKDIV0	RW	0x300	System Tick Timer divider for CPU0.	0x4000000	4.5.43
TRACECLKDIV	RW	0x308	TRACE clock divider.	0x4000000	4.5.44
CANCLKDIV	RW	0x30C	Selects the CAN clock divider.	0x0	4.5.45
FLEXFRG0CTRL	RW	0x320	Fractional Rate Divider for Flexcomm Interface 0.	0x0	4.5.46
FLEXFRG1CTRL	RW	0x324	Fractional Rate Divider for Flexcomm Interface 1.	0x0	4.5.46
FLEXFRG2CTRL	RW	0x328	Fractional Rate Divider for Flexcomm Interface 2.	0x0	4.5.46
FLEXFRG3CTRL	RW	0x32C	Fractional Rate Divider for Flexcomm Interface 3.	0x0	4.5.46

Table 39. Register overview: SYSCON (base address = 0x50000000) bit description *...continued*

Name	Access	Offset	Description	Reset value	Section
FLEXFRG4CTRL	RW	0x330	Fractional Rate Divider for Flexcomm Interface 4.	0x0	4.5.46
FLEXFRG5CTRL	RW	0x334	Fractional Rate Divider for Flexcomm Interface 5.	0x0	4.5.46
FLEXFRG6CTRL	RW	0x338	Fractional Rate Divider for Flexcomm Interface 6.	0x0	4.5.46
FLEXFRG7CTRL	RW	0x33C	Fractional Rate Divider for Flexcomm Interface 7.	0x0	4.5.46
AHBCLKDIV	RW	0x380	System clock divider.	0x0	4.5.47
CLKOUTDIV	RW	0x384	CLKOUT clock divider.	0x4000000	4.5.48
FROHFDIV	RW	0x388	FRO_HF. FRO_HF clock is the output of the FRO_192 divided by 2 (96 MHz).	0x4000000	4.5.49
WDTCLKDIV	RW	0x38C	WDT clock divider.	0x4000000	4.5.50
ADCCLKDIV	RW	0x394	ADC clock divider.	0x4000000	4.5.51
MCLKDIV	RW	0x3AC	I^{2}S MCLK clock divider.	0x4000000	4.5.52
SCTCLKDIV	RW	0x3B4	SCT/PWM clock divider.	0x4000000	4.5.53
PLL0CLKDIV	RW	0x3C4	PLL0 clock divider.	0x4000000	4.5.54
CLOCKGENUPDATELOCKOUT	RW	0x3FC	Control clock configuration registers access.	0x0	4.5.55
FMCCR	RW	0x400	FMC configuration register.	0x0	4.5.56
FMCFLUSH	W	0x41C	FMC flush control.	0x0	4.5.57
MCLKIO	RW	0x420	MCLK control.	0x0	4.5.58
FLASHREMAP_SIZE	RW	0x440	This 32-bit register contains the size of the image to remap, in bytes. The 12 LSBs are ignored, so the size granularity is 4KB.	0x0	4.5.59
FLASHREMAP_SIZE_DP	RW	0x444	This 32-bit register is a duplicate of FLASHREMAPSIZE for increased security.	0x0	4.5.59
FLASHREMAP_OFFSET	RW	0x448	This 32-bit register contains the offset by which the image is to be remapped. The 12 LSBs are ignored, so the remap granularity is 4KB.	0x0	4.5.59
FLASHREMAP_OFFSET_DP	RW	0x44C	This 32-bit register is a duplicate of FLASHREMAPOFFSET for increased security.	0x0	4.5.59
FLASHREMAP_LOCK	RW	0x45C	Control write access to FLASHREMAP_SIZE and FLASHREMAP_OFFSET registers.	0x0	4.5.59
CASPER_CTRL	RW	0x470	Controls CASPER integration.	undefined	4.5.60
PLL1CTRL	RW	0x560	PLL1 550m control.	0x0	4.5.61
PLL1STAT	R	0x564	PLL1 550m status.	0x0	4.5.61
PLL1NDEC	RW	0x568	PLL1 550m N divider.	0x0	4.5.61
PLL1MDEC	RW	0x56C	PLL1 550m M divider.	0x0	4.5.61
PLL1PDEC	RW	0x570	PLL1 550m P divider.	0x0	4.5.61
PLL0CTRL	RW	0x580	PLL0 550m control.	0x0	4.5.62

Table 39. Register overview: SYSCON (base address = 0x50000000) bit description *…continued*

Name	Access	Offset	Description	Reset value	Section
PLL0STAT	R	0x584	PLL0 550m status.	0x0	4.5.62
PLL0NDEC	RW	0x588	PLL0 550m N divider.	0x0	4.5.62
PLL0PDEC	RW	0x58C	PLL0 550m P divider.	0x0	4.5.62
PLL0SSCG0	RW	0x590	System PLL Spread Spectrum Wrapper control register 0.	0x0	4.5.62
PLL0SSCG1	RW	0x594	System PLL Spread Spectrum Wrapper control register 1.	0x0	4.5.62
FUNCRETENTIONCTRL [1]	RW	0x704	Functional retention control register.	0x0050C0000	4.5.63
CPSTAT	R	0x80C	CPU Status.	0x0	4.5.64
DICE_REG0	RW	0x900	Can be used as scratch register.	0x0	4.5.65
DICE_REG1	RW	0x904	Can be used as scratch register.	0x0	4.5.65
DICE_REG2	RW	0x908	Can be used as scratch register.	0x0	4.5.65
DICE_REG3	RW	0x90C	Can be used as scratch register.	0x0	4.5.65
DICE_REG4	RW	0x910	Can be used as scratch register.	0x0	4.5.65
DICE_REG5	RW	0x914	Can be used as scratch register.	0x0	4.5.65
DICE_REG6	RW	0x918	Can be used as scratch register.	0x0	4.5.65
DICE_REG7	RW	0x91C	Can be used as scratch register.	0x0	4.5.65
BOOT_SEED_REG0	RW	0x920	Boot seed for random number generator.	0x0	4.5.66
BOOT_SEED_REG1	RW	0x924	Boot seed for random number generator.	0x0	4.5.66
BOOT_SEED_REG2	RW	0x928	Boot seed for random number generator.	0x0	4.5.66
BOOT_SEED_REG3	RW	0x92C	Boot seed for random number generator.	0x0	4.5.66
BOOT_SEED_REG4	RW	0x930	Boot seed for random number generator.	0x0	4.5.66
BOOT_SEED_REG5	RW	0x934	Boot seed for random number generator.	0x0	4.5.66
BOOT_SEED_REG6	RW	0x938	Boot seed for random number generator.	0x0	4.5.66
BOOT_SEED_REG7	RW	0x93C	Boot seed for random number generator.	0x0	4.5.66
HMAC_REG0	RW	0x940	Control access to HMAC register.		4.5.67
HMAC_REG1	RW	0x944	Control access to HMAC register.		4.5.67
HMAC_REG2	RW	0x948	Control access to HMAC register.		4.5.67
HMAC_REG3	RW	0x94C	Control access to HMAC register.		4.5.67
HMAC_REG4	RW	0x950	Control access to HMAC register.		4.5.67
HMAC_REG5	RW	0x954	Control access to HMAC register.		4.5.67
HMAC_REG6	RW	0x958	Control access to HMAC register.		4.5.67
HMAC_REG7	RW	0x95C	Control access to HMAC register.		4.5.67
BOOT_LOCK	RW	0x960	Controls access for boot seed security.	0x0	4.5.68
CLOCK_CTRL	RW	0xA18	Controls various system clocks.	0x0	4.5.69
COMP_INT_CTRL	RW	0xB10	Comparator interrupt control.	0x0	4.5.70
COMP_INT_STATUS	RO	0xB14	Comparator interrupt status.	0x0	4.5.71
AUTOCLKGATEOVERRIDE	RW	0xE04	Control automatic clock gating.	0xFFFF	4.5.72
GPIOPSYNC	RW	0xE08	Enable bypass of the first stage of synchronization inside GPIO_INT module.	0x0	4.5.73

Table 39. Register overview: SYSCON (base address = 0x50000000) bit description *...continued*

Name	Access	Offset	Description	Reset value	Section
HASHRESTHWKEY	RW	0xF88	Controls whether the HASH AES hardware secret key is restricted to use by secure code.	0x0	4.5.74
DEBUG_LOCK_EN	RW	0xFA0	Control write access to security registers.	0x0	4.5.75
DEBUG_FEATURES	RW	0xFA4	Cortex M33 (CPU0) debug features control.	0x0	4.5.76
DEBUG_FEATURES_DP	RW	0xFA8	Cortex M33 (CPU0) debug features control DUPLICATE register.	0x0	4.5.77
SWD_ACCESS_CPU0	RW	0xFB4	Enable SWD debug access for CPU0.	0x0	4.5.78
KEY_BLOCK	W	0xFBC	Block access to PUF indexes.	0x0	4.5.79
DEBUG_AUTH_BEACON	RW	0xFC0	Debug authentication BEACON register.	0x0	4.5.80
DEVICE_ID0	R	0xFF8	Device ID register.	0x0	4.5.81
DIEID	R	0xFFC	Chip revision ID and number.	0x0	4.5.82

[1] 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.

4.5.1 Memory remap control register

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

Table 40. Memory remap control register (MEMORYREMAP, offset = 0x0) bit description

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 that 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 41. AHB Matrix priority control register (AHBMATPRIO, offset = 0x10) bit description

Bit	Symbol	Value	Description	Reset value
1:0	PRI_CPU0_CBUS		CPU0 C-AHB bus.	0x0
3:2	PRI_CPU0_SBUS		CPU0 S-AHB bus.	0x0
5:4	PRI_SDMA0		SDMA.	0x0

Table 41. AHB Matrix priority control register (AHBMATPRIO, offset = 0x10) bit description

Bit	Symbol	Value	Description	Reset value
7:6	PRI_SDMA1		SDMA secure mode.	0x0
15:8	-		Reserved. Read value is undefined, only zero should be written.	undefined
17:16	PRI_HASH_AES		HASH_AES.	0x0
19:18	PRI_CAN-FD		CAN-FD.	0x0
31:20	-		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 28 “LPC55S0x/LPC550x System Tick Timer”.

Table 42. System tick calibration for secure part of CPU0 (CPU0STCKCAL, offset = 0x38) bit description

Bit	Symbol	Value	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 28 “LPC55S0x/LPC550x System Tick Timer”.

Table 43. System tick calibration for non-secure part of CPU0 (CPU0NSTCKCAL, offset = 0x3C) bit description

Bit	Symbol	Value	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 NMI source selection register

The NMI source selection register selects a peripheral interrupts as source for the NMI interrupt. For a list of all peripheral interrupts and their IRQ numbers, see Table 44.

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 description

Bit	Symbol	Value	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
30:6	-		Reserved. Read value is undefined, only zero should be written.	undefined
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.

4.5.6 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 description

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.	
6:5	-		Reserved. Read value is undefined, only zero should be written.	undefined
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

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

Bit	Symbol	Value	Description	Reset value
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
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

4.5.7 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: To avoid the unintentional setting or clearing of other bits, It is recommended that changes to the PRESETCTRL registers be accomplished by using the related PRESETCTRLSET and PRESETCTRLCLR registers.

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

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.	
6:3	-		Reserved. Read value is undefined, only zero should be written.	undefined
7	CAN_RST		CAN reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
9:8			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.	
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.	

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

Bit	Symbol	Value	Description	Reset value
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.	
25:23	-		Reserved. Read value is undefined, only zero should be written.	undefined
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

4.5.8 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: To avoid the unintentional setting or clearing of other bits, It is recommended that changes to the PRESETCTRL registers be accomplished by using the related PRESETCTRLSET and PRESETCTRLCLR registers.

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

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.	
5:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

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

Bit	Symbol	Value	Description	Reset value
6	SRAM_CTRL3_RST		SRAM Controller 3 reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
7	-		Reserved. Read value is undefined, only zero should be written.	undefined
8	FREQME_RST		Frequency meter 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	CWT_RST		Code Watchdog 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	RNG_RST		RNG reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
14	-		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.	
17:16	-		Reserved. Read value is undefined, only zero should be written	undefined
18	HASH_AES_RST		HASH AES reset control.	0x0
		1	Block is reset.	
		0	Block is not reset.	
19	-		Reserved. Read value is undefined, only zero should be written.	undefined
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

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

Bit	Symbol	Value	Description	Reset value
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.

4.5.9 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 description

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

4.5.10 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 description.

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

4.5.11 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 description

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

4.5.12 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 description

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

4.5.13 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 description

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

4.5.14 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 description

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

4.5.15 Software reset register

Write 0x5A00_0001 to generate a software reset.

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

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	

4.5.16 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, APB bridges, CPU, SYSCON block, and 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.	
6:5	-			Reserved.	
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
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.	
17:16	-	WO		Reserved. Read value is undefined, only zero should be written.	undefined
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.	

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

Bit	Symbol	Access	Value	Description	Reset value
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.	
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

4.5.17 AHB clock control 1

The AHBCLKCTRL1 register enables the clocks to individual peripheral blocks.

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

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.	
5:3	-		Reserved. Read value is undefined, only zero should be written.	undefined
6	-		Reserved. Read value is undefined, only zero should be written.	0x0
7	CAN		Enables the clock for the CAN.	0x0
		1	Enable clock.	
		0	Disable clock.	

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

Bit	Symbol	Value	Description	Reset value
9:8	-		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.	
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.	
25:23	-		Reserved. Read value is undefined, only zero should be written.	undefined
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.	

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

Bit	Symbol	Value	Description	Reset value
28	-		Reserved. Read value is undefined, only zero should be written.	0x0
29	-		Reserved. Read value is undefined, only zero should be written.	undefined
31:30	-		Reserved. Read value is undefined, only zero should be written.	0x0

4.5.18 AHB clock control 2

The AHBCLKCTRL2 register enables the clocks to individual peripheral blocks.

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

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.	
5:3	-		Reserved. Read value is undefined, only zero should be written.	0x0
6	SRAM_CTRL3		Enables the clock for the SRAM Controller 3.	0x0
		1	Enable clock.	
		0	Disable clock.	
7	-		Reserved. Read value is undefined, only zero should be written.	0x0
8	FREQME		Enables the clock for the frequency meter.	0x0
		1	Enable clock.	
		0	Disable clock.	
10:9	-		Reserved. Read value is undefined, only zero should be written.	0x0
11	CWT		ENABLE CODE-WDG clock.	0x0
		1	Enable clock.	
		0	Disable clock.	
12	-		Reserved. Read value is undefined, only zero should be written.	0x0
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.	0x0
15	SYSCTL		SYSCTL block clock.	0x0
		1	Enable clock.	
		0	Disable clock.	
17:16	-		Reserved. Read value is undefined, only zero should be written.	undefined
18	HASH_AES		Enables the clock for the HASH_AES.	0x0
		1	Enable clock.	
		0	Disable clock.	
19	-		Reserved. Read value is undefined, only zero should be written.	0x0
20	PLULUT		Enables the clock for the PLU LUT.	0x0
		1	Enable clock.	
		0	Disable clock.	

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

Bit	Symbol	Value	Description	Reset value
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.	
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.	0x0

4.5.19 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 description

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

4.5.20 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 description

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

4.5.21 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 description

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

4.5.22 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 description

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

4.5.23 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 description

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

4.5.24 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 description

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

4.5.25 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 description

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

4.5.26 Trace clock source select register

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

Table 65. Trace clock source select (TRACECLKSEL, offset = 0x268) bit description

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

4.5.27 CTimer 0 clock source select

This register selects the clock source for CTimer 0.

Table 66. CTimer 0 clock source select (CTIMERCLKSEL0, offset = 0x26C) bit description

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

4.5.28 CTimer 1 clock source select register

This register selects the clock source for CTimer 1.

Table 67. CTimer 1 clock source select (CTIMERCLKSEL1, offset = 0x270) bit description

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

4.5.29 CTimer 2 clock source select register

This register selects the clock source for the CTimer 2.

Table 68. CTimer 2 clock source select (CTIMERCLKSEL2, offset = 0x274) bit description

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

4.5.30 CTimer 3 clock source select register

This register selects the clock source for CTimer 3.

Table 69. CTimer 3 clock source select (CTIMERCLKSEL3, offset = 0x278) bit description

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

4.5.31 CTimer 4 clock source select register

This register selects the clock source for CTimer 4.

Table 70. CTimer 4 clock source select (CTIMERCLKSEL4, offset = 0x27C) bit description

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

4.5.32 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 72, 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 71. Main clock A source select (MAINCLKSELA, offset=0x280) bit description.

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

4.5.33 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 71, 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 72. Main clock B source select (MAINCLKSELB, offset = 0x284) bit description

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

4.5.34 CLKOUT clock source select register A

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

Table 73. CLKOUT clock source select (CLKOUTSEL, offset = 0x288) bit description

Bit	Symbol	Value	Description	Reset value
3:0	SEL		CLKOUT clock source select.	0xF
		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.	

Table 73. CLKOUT clock source select (CLKOUTSEL, offset = 0x288) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
		8	Reserved.	
		9	Reserved.	
		10	Reserved.	
		11	Reserved.	
		12	No clock.	
		13	No clock.	
		14	No clock.	
		15	No clock.	
31:4	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.35 PLL0 clock source select register

This register selects the clock source for the PLL0.

Table 74. PLL0 clock source select (PLL0CLKSEL, offset = 0x290) bit description

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

4.5.36 PLL1 clock source select register

This register selects the clock source for PLL1.

Table 75. PLL1 clock source select (PLL1CLKSEL, offset = 0x294) bit description

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

4.5.37 CAN clock select

This register selects the CAN clock.

Table 76. CAN clock source select (CANCLKSEL, offset 0x2A0)

Bit	Symbol	Value	Description	Reset value
2:0	SEL		CAN clock source select.	0x7
		0	CAN 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

4.5.38 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 description

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	Xtal clock coming directly.	
		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 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 78. Flexcomm Interface 0 clock source select for Fractional Rate Divider (FCCLKSEL0, offset = 0x2B0) bit description

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 79. Flexcomm Interface 1 clock source select for Fractional Rate Divider (FCCLKSEL1, offset = 0x2B4) bit description

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 80. Flexcomm Interface 2 clock source select for Fractional Rate Divider (FCCLKSEL2, offset = 0x2B8) bit description

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 81. Flexcomm Interface 3 clock source select for Fractional Rate Divider (FCCLKSEL3, offset = 0x2BC) bit description

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 82. Flexcomm Interface 4 clock source select for Fractional Rate Divider (FCCLKSEL4, offset = 0x2C0) bit description

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 83. Flexcomm Interface 5 clock source select for Fractional Rate Divider (FCCLKSEL5, offset = 0x2C4) bit description

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 84. Flexcomm Interface 6 clock source select for Fractional Rate Divider (FCCLKSEL6, offset = 0x2C8) bit description

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 85. Flexcomm Interface 7 clock source select for Fractional Rate Divider (FCCLKSEL7, offset = 0x2CC) bit description

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

4.5.40 HS SPI clock source select register

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

Table 86. HS SPI clock source select (HSLSPICLKSEL, offset = 0x2D0) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		HS LSPI 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

4.5.41 I^{2}S MCLK clock source select register

This register selects a clock to provide to the I^{2}S MCLK output function. In a system using I^{2}S and/or digital microphone, this should be related to the clock used by those functions.

Table 87. I^{2}S MCLK clock source select (MCLKCLKSEL, offset = 0x2E0) bit description

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

4.5.42 SCTimer/PWM clock source select register

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

Table 88. SCTimer/PWM clock source select (SCTCLKSEL, offset = 0x2F0) bit description

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

4.5.43 SYSTICK clock divider register 0

This register configures the SYSTICK divider clock for CPU0.

Table 89. System Tick Timer divider for CPU0 (SYSTICKCLKDIV0, offset = 0x300) bit description

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.	

4.5.44 Trace clock divider register

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

Table 90. TRACE clock divider (TRACECLKDIV, offset = 0x308) bit description

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.	

4.5.45 CAN clock divider

This register selects the CAN clock divider.

Table 91. CAN clock divider (CANCLKDIV, offset 0x30C) bit description

Bit	Symbol	Value	Description	Reset value
7:0	DIV		Clock divider value. 0: divide by 1 to 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. Read only.	0x0
		1	Clock frequency is not stable.	
		0	Clock frequency is stable	

4.5.46 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 depends on both:

-the functionality (USART, SPI, I2C or I2S) enabled in the Flexcomm,

-the maximum System Frequency that has been set up by the user.

When the maximum System Frequency chosen by the user is below or equal to 72 MHz, the FRG maximum allowed output frequency is:

- For USART, the FRG output frequency must not be higher than 25 MHz.
- For SPI, the FRG output frequency must not be higher than 33 MHz.
- For I2S, the FRG output frequency must not be higher than 25 MHz.
- For I2C, the FRG output frequency must not be higher than 20 MHz.
- For High Speed SPI, the FRG output frequency must not be higher than 72 MHz.

When the maximum System Frequency chosen by the user is above 72 MHz, the FRG maximum allowed output frequency is:

- For USART, the FRG output frequency must not be higher than 100 MHz.
- For SPI, the FRG output frequency must not be higher than 100 MHz.
- For I2S, the FRG output frequency must not be higher than 100 MHz.
- For I2C, the FRG output frequency must not be higher than 100 MHz.
- For High Speed SPI, the FRG output frequency must not be higher than 100 MHz.

Table 92. FRG maximum allowed frequency

	FCLK max. [Maximum FRG output frequency, MHz]	
	Max. System Frequency <= 72 MHz	Max. System Frequency > 72 MHz
USART	25	100
SPI	33	100
I2C	20	100
I2S	25	100
High Speed SPI	72	100

The output rate is:

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 4.5.39 “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 93. Fractional rate divider for Flexcomm Interface 0 (FLEXFRG0CTRL, offset = 0x320) bit description

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 94. Fractional rate divider for Flexcomm Interface 1 (FLEXFRG1CTRL, offset = 0x324) bit description

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 2 (FLEXFRG2CTRL, offset = 0x328) bit description

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 3 (FLEXFRG3CTRL, offset = 0x32C) bit description

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 4 (FLEXFRG4CTRL, offset = 0x330) bit description

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 5 (FLEXFRG5CTRL, offset = 0x334) bit description

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 6 (FLEXFRG6CTRL, offset = 0x338) bit description

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 7 (FLEXFRG7CTRL, offset = 0x33C) bit description

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

4.5.47 AHB clock divider register

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

Table 101. System clock divider (AHBCLKDIV, offset = 0x380) bit description

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.	

4.5.48 CLKOUT clock divider register

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

Table 102. CLKOUT clock divider (CLKOUTDIV, offset = 0x384) bit description

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 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 103. FRO_HF clock divider (FROHFDIV, offset = 0x388) bit description

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.	

4.5.50 WWDT clock divider

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

Table 104. WDT clock divider (WDTCLKDIV, offset = 0x38C) bit description

Bit	Symbol	Value	Description	Reset value
5:0	DIV		Clock divider value. 0: Divide by 1. … 255: Divide by 256.	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.	

4.5.51 ADC clock source divider register

This register divides the clock to the ADC.

Table 105. ADC clock divider (ADCCLKDIV, offset = 0x394) bit description

Bit	Symbol	Value	Description	Reset value
2:0	DIV		Clock divider value. 0: Divide by 1. … 255: Divide by 256	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.	

4.5.52 I^{2}S MCLK clock divider register

This register determines the divider value for the I^{2}S MCLK output, if used by the application.

Table 106. I^{2}S MCLK clock divider (MCLKDIV, offset = 0x3AC) bit description

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 SCTimer/PWM clock divider

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

Table 107. SCTimer/PWM clock divider (SCTCLKDIV, offset = 0x3B4) bit description

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.	

4.5.54 PLL0 clock divider

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

Table 108. PLL0 clock divider (PLL0CLKDIV, offset = 0x3C4) bit description

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.	

4.5.55 Control clock configuration registers access

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

Table 109. Control clock configuration registers access (xxxDIV, xxxSEL) (CLOCKGENUPDATELOCKOUT, offset = 0x3FC) bit description

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.	

4.5.56 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 through software as shown in Table 110.

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 110. 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]	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.	
11:7	-	-	Reserved.	-
15:12	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 84 MHz).	
		0x7	8 system clocks flash access time (for system clock rates up to 96 MHz).	
31:16	-	-	Reserved.	-

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

4.5.57 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 111. FMC flush control (FMCFLUSH, offset = 0x41C) bit description

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

4.5.58 MCLKIO control

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

Table 112. MCLK control (MCLKIO, offset = 0x420) bit description

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

4.5.59 Flash Remap

The values of FLASHREMAPSIZE and FLASHREMAPSIZE_DP must be set to the same value for their settings to become effective. Any time FLASHREMAPSIZE != FLASHREMAPSIZE_DP, a read of the flash via the FMC will result in an abort response.When both FLASHREMAPSIZE and FLASHREMAPOFFSET are set to non-zero values, a flash read within the AHB address range 0x0 to FLASHREMAPSIZE has its addresses incremented by FLASHREMAPOFFSET and then presented to the flash. This effectively allows an image starting at address FLASHREMAPOFFSET to appear to be at address 0x0.

Software must flush FMC buffers after changing the flash remap settings by writing to the FMCFLUSH register. The condition FLASHREMAPOFFSET >= FLASHREMAPSIZE must be met, otherwise Image 0 and Image 1 will overlap. When Image 1 is “active”, Image 0 is not visible for reads via the FMC. Flash remapping only affects FMC read addresses and has no effect on programming addresses. When programming, the physical (non-remapped) addresses should be used.

The PRINCE always operates off the physical address. So, the physical (non-remapped) addresses of the code should be used when configuring the PRINCE regions. Security attributes operate off the remapped AHB address. Software is responsible to make any adjustments required to security attributes when flash address remap settings are changed.

Table 113. This 32-bit register contains the offset by which the image is to be remapped (FLASHREMAP_SIZE, offset 0x440)

Bit	Symbol	Value	Description	Reset value
31:0	FLASHREMAP_SIZE		This 32-bit register contains the size of the image to remap, in bytes. The 12 LSBs are ignored, so the size granularity is 4KB.	0x0

Table 114. This 32-bit register is a duplicate of FLASHREMAPSIZE for increased security (FLASHREMAP_SIZE_DP, offset 0x444)

Bit	Symbol	Value	Description	Reset value
31:0	FLASHREMAP_SIZE_DP		The values of FLASHREMAPSIZE and FLASHREMAPSIZE_DP must be set to the same value for their settings to become effective. Whenever time FLASHREMAPSIZE != FLASHREMAPSIZE_DP, a read of the flash via the FMC will result in an abort response.	0x0

Table 115. This 32-bit register contains the offset by which the image is to be remapped (FLASHREMAP_OFFSET, offset 0x448)

Bit	Symbol	Value	Description	Reset value
31:0	FLASHREMAP_OFFSET		This 32-bit register contains the offset by which the image is to be remapped. The 12 LSBs are ignored, so the remap granularity is 4KB. (FLASHREMAP_OFFSET).	0x0

Table 116. This 32-bit register is a duplicate of FLASHREMAPOFFSET for increased security (FLASHREMAP_OFFSET_DP, offset 0x44C)

Bit	Symbol	Value	Description	Reset value
31:0	FLASHREMAP_OFFSET_DP		The values of FLASHREMAPOFFSET and FLASHREMAPOFFSET_DP must be set to the same value for their settings to become effective. Any time FLASHREMAPOFFSET != FLASHREMAPOFFSET_DP, a read of the flash via the FMC will result in an abort response.	0x0

Table 117. Control write access to FLASHREMAP_SIZE and FLASHREMAP_OFFSET registers (FLASHREMAP_LOCK, offset 0x45C)

Bit	Symbol	Value	Description	Reset value
31:0	LOCK		Control write access to FLASHREMAP_SIZE and FLASHREMAP_OFFSET registers. Any value other than 0xC33CA55A and 0x3CC35AA5 does not modify the state.	0xC33CA 55A
		101943 5685	Write access to 4 registers FLASHREMAP_SIZEand FLASHREMAP_OFFSETis unlocked.	
		327553 1610	Write access to 4 registers FLASHREMAP_SIZEand FLASHREMAP_OFFSETis locked.	

4.5.60 CASPER

Controls CASPER integration.

Table 118. Control CASPER integration. (CASPER_CTRL, offset 0x470)

Bit	Symbol	Value	Description	Reset value
0	INTERLEAVE		Control RAM access for RAMX0 and RAMX1.	0x0
		1	RAM access to RAMX0 and RAMX1 is interleaved.	
		0	RAM access to RAMX0 and RAMX1 is consecutive.	
31:1	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.61 PLL1 Registers

4.5.61.1 PLL1 control register

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

Table 119. PLL1 550m 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

Table 119. PLL1 550m control (PLL1CTRL, offset = 0x560) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
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

4.5.61.2 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 120. PLL1 status register (PLL1STAT, offset = 0x564)

Bit	Symbol	Access	Value	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

4.5.61.3 PLL1 N-divider register

The PLL1NDEC controls operation of the PLL pre-divider.

Table 121. PLL1 N divider (PLL1NDEC, offset = 0x568)

Bit	Symbol	Access	Value	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

4.5.61.4 PLL1 M-divider register

The PLL1MDEC controls operation of the PLL feedback divider.

Table 122. PLL1 M divider (PLL1MDEC, offset = 0x56C)

Bit	Symbol	Access	Value	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

4.5.61.5 PLL1 P-divider register

The PLL1PDEC controls operation of the PLL post-divider.

Table 123. PLL1 P divider (PLL1PDEC, offset = 0x570)

Bit	Symbol	Access	Value	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

4.5.62 PLL0 Registers

4.5.62.1 PLL0 control register

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

Table 124. PLL0 550m control (PLL0CTRL, offset = 0x580) bit description

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.	

Table 124. PLL0 550m control (PLL0CTRL, offset = 0x580) bit description *…continued*

Bit	Symbol	Access	Value	Description	Reset value
23	FRMCLKSTABLE	RW		Free running mode clock stable. Note: frm_clockstable can be =1 only after the PLL output frequency is stable.	0x0
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.	undefi ned

4.5.62.2 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 125. PLL0 550m status (PLL0STAT, offset = 0x584) bit description

Bit	Symbol	Access	Value	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: Refer to the PLL Lock Bit Errata regarding details on ensuring the PLL is stable and locked.

4.5.62.3 PLL0 N-divider register

The PLL0NDEC controls operation of the PLL pre-divider.

Table 126. PLL0 550m N divider (PLL0NDEC, offset = 0x588) bit description

Bit	Symbol	Access	Value	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

4.5.62.4 PLL0 P-divider register

The PLL0PDEC controls operation of the PLL post-divider.

Table 127. PLL0 550m P divider (PLL0PDEC, offset = 0x58C) bit description

Bit	Symbol	Access	Value	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

4.5.62.5 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.

PLL0 spread spectrum control register 0

Table 128. PLL0 spread spectrum wrapper control register 0 (PLL0SSCG0, offset = 0x590) bit description

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 129. PLL0 spread spectrum wrapper control register 1 (PLL0SSCG1, offset = 0x594) bit description

Bit	Symbol	Access	Value	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$^{\text{kss/Fcco}}$= 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

4.5.63 Functional retention 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 function retention control register specifies retention of data in SRAM during power down.

Table 130. Functional retention control (FUNCRTETENTIONCTRL, offset = 0x704) bit description

Bit	Symbol	Access	Value	Description	Reset value
0	FUNCRETENA	RW		Functional retention in power down only.	0x0
			1	Enable functional retention.	0x0
			0	Disable functional retention.	0x0

Table 130. Functional retention control (FUNCRTETENTIONCTRL, offset = 0x704) bit description

Bit	Symbol	Access	Value	Description	Reset value
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	-	W0	-	Reserved. Read value is undefined, only zero should be written.	undefined

4.5.64 CPU status

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

Table 131. CPU status (CPSTAT, offset = 0x80C) bit description

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	-		Reserved.	undefined
2	CPU0LOCKUP		The CPU0 lockup state.	0x0
		1	The CPU is in lockup.	
		0	The CPU is not in lockup.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.65 DICE register

LPC55S0x 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 132. (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

4.5.66 Boot seed

Provides access to boot seed registers for generating random numbers.

Table 133. boot seed (256-bit random value) (BOOT_SEED_REG0, offset 0x920)

Bit	Symbol	Value	Description	Reset value
31:0	BOOT_SEED_REG0		Boot seed register.	0x0

Table 134. boot seed (256-bit random value) (BOOT_SEED_REG1, offset 0x924)

Bit	Symbol	Value	Description	Reset value
31:0	BOOT_SEED_REG1		Boot seed register.	0x0

Table 135. boot seed (256-bit random value) (BOOT_SEED_REG2, offset 0x928)

Bit	Symbol	Value	Description	Reset value
31:0	BOOT_SEED_REG2		Boot seed register.	0x0

Table 136. boot seed (256-bit random value) (BOOT_SEED_REG3, offset 0x92C)

Bit	Symbol	Value	Description	Reset value
31:0	BOOT_SEED_REG3		Boot seed register.	0x0

Table 137. boot seed (256-bit random value) (BOOT_SEED_REG4, offset 0x930)

Bit	Symbol	Value	Description	Reset value
31:0	BOOT_SEED_REG4		Boot seed register.	0x0

Table 138. boot seed (256-bit random value) (BOOT_SEED_REG5, offset 0x934)

Bit	Symbol	Value	Description	Reset value
31:0	BOOT_SEED_REG5		Boot seed register.	0x0

Table 139. boot seed (256-bit random value) (BOOT_SEED_REG6, offset 0x938)

Bit	Symbol	Value	Description	Reset value
31:0	BOOT_SEED_REG6		Boot seed register.	0x0

Table 140. boot seed (256-bit random value) (BOOT_SEED_REG7, offset 0x93C)

Bit	Symbol	Value	Description	Reset value
31:0	BOOT_SEED_REG7		Boot seed register.	0x0

4.5.67 HMAC

Provides access to HMAC.

Table 141. HMAC (HMAC_REG0, offset 0x940)

Bit	Symbol	Value	Description	Reset value
31:0	HMAC_REG0		Boot seed register.	0x0

Table 142. HMAC (HMAC_REG1, offset 0x944)

Bit	Symbol	Value	Description	Reset value
31:0	HMAC_REG1		Boot seed register.	0x0

Table 143. HMAC (HMAC_REG2, offset 0x948)

Bit	Symbol	Value	Description	Reset value
31:0	HMAC_REG2		Boot seed register.	0x0

Table 144. HMAC (HMAC_REG3, offset 0x94C)

Bit	Symbol	Value	Description	Reset value
31:0	HMAC_REG3		Boot seed register.	0x0

Table 145. HMAC (HMAC_REG4, offset 0x950)

Bit	Symbol	Value	Description	Reset value
31:0	HMAC_REG4		Boot seed register.	0x0

Table 146. HMAC (HMAC_REG5, offset 0x954)

Bit	Symbol	Value	Description	Reset value
31:0	HMAC_REG5		Boot seed register.	0x0

Table 147. HMAC (HMAC_REG6, offset 0x958)

Bit	Symbol	Value	Description	Reset value
31:0	HMAC_REG6		Boot seed register.	0x0

Table 148. HMAC (HMAC_REG7, offset 0x95C)

Bit	Symbol	Value	Description	Reset value
31:0	HMAC_REG7		Boot seed register.	0x0

4.5.68 Control write access to boot seed

Controls write access to the boot seed.

Table 149. Control write access to boot seed security registers (BOOT_LOCK, offset 0x960)

Bit	Symbol	Access	Value	Description	Reset value
0	LOCK_BOOT_SEED	RW		Control write access to BOOT_SEED_REG registers	0x0
			1	Write access to all 8 registers BOOT_SEED_REG is locked. This register is write once.	
1	LOCK_HMAC	RW		Control write access to HMAC_REG registers.	0x0
			1	Write access to all 8 registers HMAC_REG is locked. This register is write once.	
31:2	-	WO		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.69 Clock control

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

Table 150. Various system clock controls (CLOCK_CTRL, offset = 0xA18) bit description

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.	
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.	

Table 150. Various system clock controls (CLOCK_CTRL, offset = 0xA18) bit description

Bit	Symbol	Access	Value	Description	Reset value
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

4.5.70 Comparator interrupt control

This register is to control the interrupt handler for comparator.

Table 151. Comparator interrupt control (COMP_INT_CTRL, offset = 0xB10) bit description

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

4.5.71 Comparator interrupt status

This register indicates comparator interrupt status.

Table 152. Comparator interrupt status (COMP_INT_STATUS, offset = 0xB14) bit description

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

4.5.72 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 153. Control automatic clock gating (AUTOCLKGATEOVERRIDE, offset = 0xE04) bit description

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	
6:5	-		Reserved.	

Table 153. Control automatic clock gating (AUTOCLKGATEOVERRIDE, offset = 0xE04) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
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	
14	-		Reserved. Must be written with 1. Ignored upon READ.	0x1
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
		49374	Bit Fields 0 - 15 of this register are updated.	
		0	Any other values than 0xC0DE. Bit Fields 0 - 15 of this register are not updated.	

4.5.73 Enable bypass of the first stage

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

Table 154. Control of synchronization inside GPIO_INT module (GPIOPSYNC, offset = 0xE08) bit description

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 GPIO_INT module.	
		0	Use the first stage of synchronization inside GPIO_INT module.	
31:1	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.74 Restrict HASH AES hardware secret key

This register controls whether the HASH AES hardware secret key is restricted to use by secure code.

Table 155. Controls HASH AES hardware secret key restrictions (HASHRESTHWKEY, offset 0xF88) bit description

Bit	Symbol	Value	Description	Reset value
31:0	UNLOADCODE	3275531610	Code value that controls whether HASH AES hardware secret key is unlocked HASH AES hardware secret key is unlocked for use by non-secure code. Any other value means that the hardware secret key is restricted to use by secure code only.	0x0

4.5.75 Debug lock enable

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

Table 156. Debug Lock Enable (DEBUG_LOCK_EN, offset = 0xFA0) bit description

Bit	Symbol	Value	Description	Reset value
3:0	LOCK_ALL		Control write access to security registers: Control write access to CODESECURITYPROTTEST, CODESECURITYPROTCPU0, CPU0_DEBUG_FEATURES, and DBG_AUTH_SCRATCH registers.	0xA
		10	Enables write access to all six registers.	
		0	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

4.5.76 Debug features control

This register controls CPU0 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 157. Debug Features register (DEBUG_FEATURES, offset = 0xFA4) bit description

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.	
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.	
31:8	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.77 Debug features control duplicate

This register controls the CPU0 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 158. Debug Features Duplicate register (DEBUG_FEATURES_DP, offset = 0xFA8) bit description

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.	

Table 158. Debug Features Duplicate register (DEBUG_FEATURES_DP, offset = 0xFA8) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
7:6	CPU0_SPNIDEN		CPU0 secure non invasive debug control.	0x1
		0x2	Invasive debug is enabled.	
		0x1	Invasive debug is disabled.	
		Others	Reserved.	
31:8	-		Reserved. Read value is undefined, only zero should be written.	undefined

4.5.78 SWD access port for CPU0

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

Table 159. SWD access port for CPU0 (SWD_ACCESS_CPU0, offset = 0xFB4) bit description

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.	

4.5.79 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 160. Key Block register (KEY_BLOCK, offset = 0xFBC) bit description

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

4.5.80 Debug authentication BEACON register

This register is a register protected by security. ROM sets register (read only) with value received in debug credentials before passing control to user code. This can be used to extend debug authentication control for customer application. Please refer to Debug authentication section in the Debug chapter.

Table 161. Debug Authentication Scratch registers (DEBUG_AUTH_BEACON, offset = 0xFC0) bit description

Bit	Symbol	Value	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

4.5.81 Device ID register

This register describes the device ID.

Table 162. Device ID0 register (DEVICE_ID0, offset = 0xFF8) bit description

Bit	Symbol	Value	Description	Reset value
31:0	DEVICE_ID		Device/Part Identification	
	0x51000486		LPC55S06	undefined
	0x51001584		LPC55S04	undefined
	0xA1000406		LPC5506	undefined
	0xA1001504		LPC5504	undefined
	0xA1003702		LPC5502	undefined

4.5.82 Chip revision ID and N number

This register describes the Chip Number and Revision.

Table 163. Chip revision ID and number (DIEID, offset = 0xFFC) bit description

Bit	Symbol	Value	Description	Reset value
3:0	REV_ID		Revision.	0x1
23:4	MCO_NUM_IN_DIE_ID		Chip number.	0x426B
31:24	-		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.

The matrix in Table 164 “Resets” 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:

Table 164. 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	CDOGRESET	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	Act
	OSTIMER	Rst	Rst	Rst	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act
	RTC	Rst	Act	Rst	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act
System Domain	IOCON	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act	Act
	SYSCON	Rst	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	Act
	PUF key management	Rst	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	Rst	Act	Act	Act
	flexcomm_3	Rst	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	Rst	Act	Act	Act
	SRAM Memory	Rst	Rst	Rst	Act	Act	Act	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	FLASH Memory	Rst	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	Rst	Act	Act
	PRINCE	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act	Act
	WDT	Rst	Rst	Rst	Rst	Rst	Act	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	UTICK	Rst	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	Rst	Act	Act
	SCT	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	cdog	Rst	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	Rst	Act	Act
	cpu0	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act	Act

Table 164. 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	CDOGRESET	Wakeup from Power Down	Wakeup from Deep Sleep	Wakeup from Sleepw
	CASPER	Rst	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	Rst	Act	Act
	RNG	Rst	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	Rst	Act	Act
	HASH	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	Secure AHB ctrl	Rst	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	Rst	Act	Act
	analog_ctrl	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	crr_d_ip_adcSAR_fifo_syn	Rst	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	Rst	Act	Act
	SDIO	Rst	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	Rst	Act	Act
	INPUTMUX	Rst	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	Rst	Act	Act
	PINT	Rst	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	Rst	Act	Act
	Secure PINT	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
	CAN	Rst	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	Rst	Act	Act
	gray decoding	Rst	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	Rst	Act	Act
	SPI filter (internal only)	Rst	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	Act	Rst	Act	Act
	Bias	Rst	Rst	Rst	Act	Act	Act	Rst	Rst	Rst	Rst	Act	Rst	Act	Act
	BOD (VBAT)	Rst	Rst	Rst	Act	Act	Act	Rst	Rst	Rst	Rst	Act	Rst	Act	Act
	BOD (CORE)	Rst	Rst	Rst	Act	Act	Act	Rst	Rst	Rst	Rst	Act	Rst	Act	Act
	LDO_AO	Rst	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act
	LDOs	Rst	Rst	Rst	Act	Act	Act	Rst	Rst	Rst	Rst	Act	Rst	Act	Act
	32 kHz XTAL	Rst	Act	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	Rst	Act	Act
	32 kHz FRO	Rst	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act	Act

Table 164. 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	CDOGRESET	Wakeup from Power Down	Wakeup from Deep Sleep	Wakeup from Sleepw
192 MHz FRO	Rst	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	Act	Rst	Act	Act
Temperature sensor	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Rst	Act	Act
ADC	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Rst	Act	Act
Analog Comparator	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Rst	Act	Act

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, or 48 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 13.4.3 “DCDC first control register [1]”.

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 “LPC55S0x/LPC550x 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 allows maximization of the performance of the CPU when it is running code from flash memory, while also saving power. The flash accelerator also provides speed and power improvements for data accesses to the flash memory.

See Section 4.5.56 “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 4 shows a simplified diagram of the flash accelerator blocks and data paths.

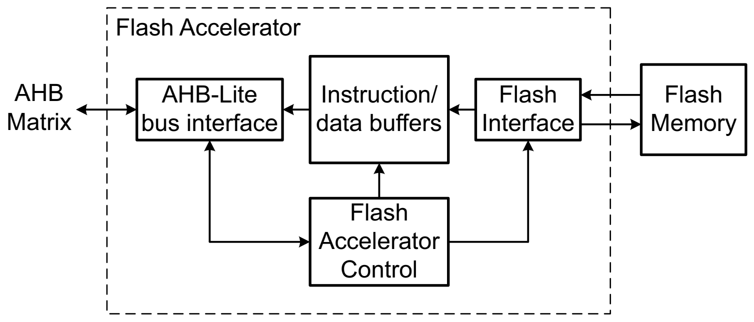


Fig 4. Simplified block diagram of the flash accelerator

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.

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 4.5.61 “PLL1 Registers”, and Section 4.5.62 “PLL0 Registers”.

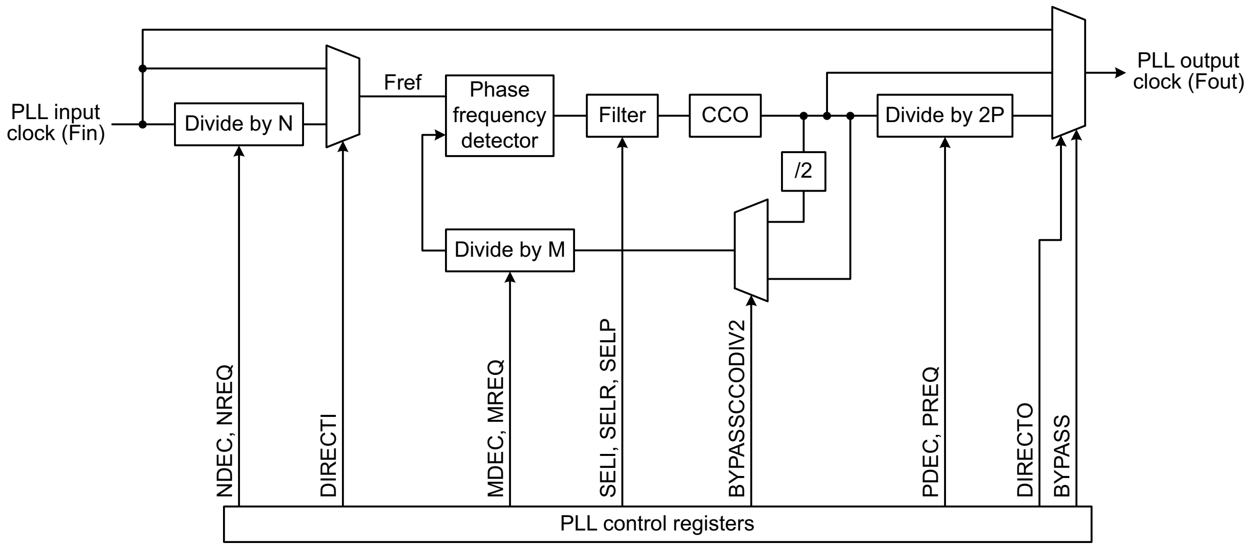


Fig 5. System PLL block diagram showing typical operation

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 2. In addition, a block diagram of the PLL is shown in Figure 5. 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 14.4.5 “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, and other peripherals. The PLL output dividers are described in the Clock Dividers section following the PLL description.

For PLL register descriptions, see Section 4.5.61 “PLL1 Registers” and Section 4.5.62 “PLL0 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.

Remark: Refer to the PLL Lock Bit Errata regarding details on ensuring the PLL is stable and locked.

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".

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 319. 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 165 and detailed in the following sections.

Table 165. 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. Note that an additional divide-by-2 may optionally be included in the divider path.
- 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 6.

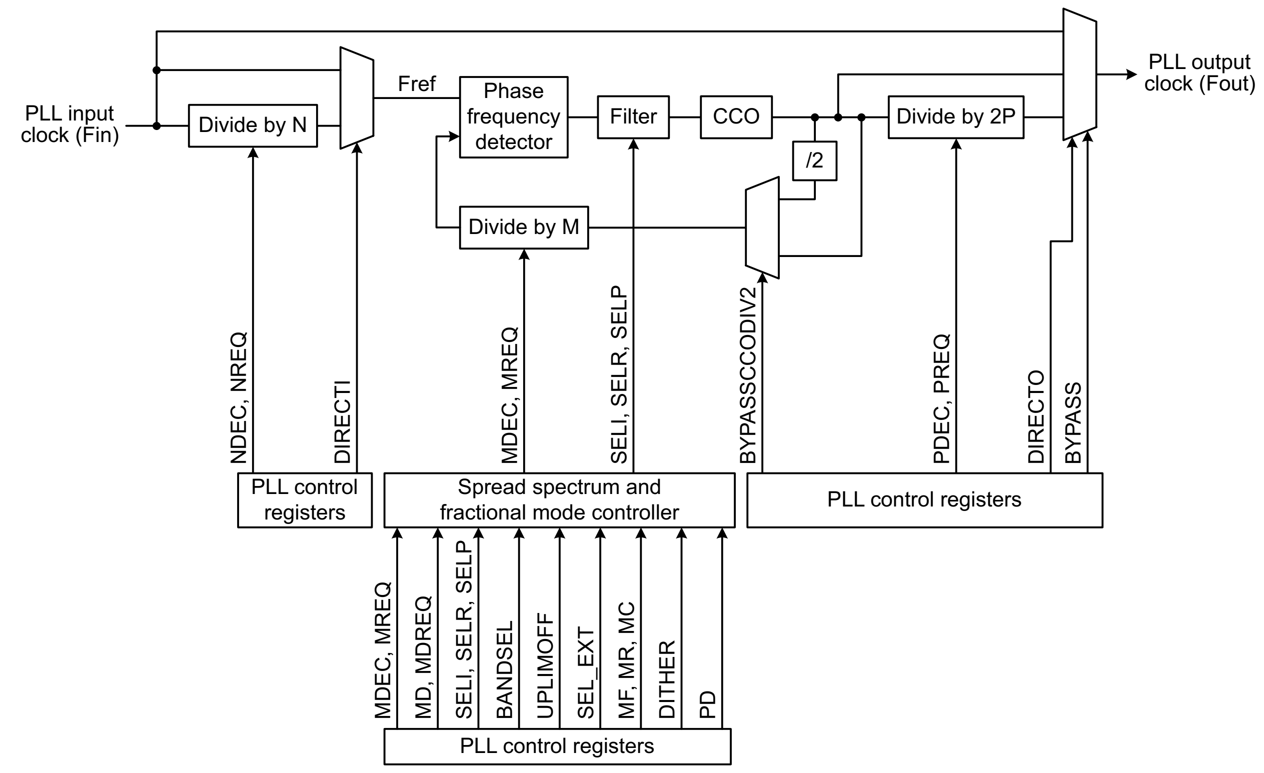


Fig 6. System PLL block diagram showing spread spectrum and fractional divide operation

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:

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

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:

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

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:

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

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:

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

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 6.

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 δfmodpk-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 $\text{directo}_{PLL} = 1$:

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

if $\text{directo}_{PLL} = 0$, $P_{PLL} = 1$:

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

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

if $\text{directo}_{PLL} = 1$:

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

if $\text{directo}_{PLL} = 0$, $P_{PLL} = 1$:

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

The spectral tone reduction/EMI reduction ΔP at F_{out} is approximately:

if $directo_{PLL}$ = 1:

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

if $directo_{PLL}$ = 0, P_{PLL} = 1:

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

See Table 166 for the spectral tone reduction and clock cycle displacement for $directo_{PLL}$ = 0 and P_{PLL}= 1.

Table 166. 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 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 4.5.61 "PLL1 Registers" and Section 4.5.62 "PLL0 Registers", and summarized below.

Table 167. Summary of PLL related registers

Register	**Description**
PLLxCTRL	PLL control.
PLLxSTAT	PLL status.
PLLxNDEC	PLL pre-divider.
PLLxPDEC	PLL post-divider.
PLL0SSCTRL	PLL spread spectrum control 0.
PLL0SSCTRL1	PLL spread spectrum control 1.

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-divide or post-divide, the LPCOpen Chip_POWER_SetPLL API can be used, see Section 14.4.5 “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 the power or accuracy that is 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 4.5.35 “PLL0 clock source select register”.
3. Set up the PLL dividers and mode settings. See Section 4.5.61 “PLL1 Registers” and Section 4.5.62 “PLL0 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 the 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 in Figure 2.

UM11424

Chapter 5: LPC55S0x/LPC550x Flash

Rev. 1.5 — 21 December 2023 User manual

5.1 General description

This chapter describes the flash controller targeted for the LPC55S0x/LPC550x device.

5.2 Features

- Includes analog delay block to manage self-timed read operations.
- 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 7 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 a 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.

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 5.6.4 “Command listing (CMD)”) will not cause a hardware fault. See AN12949 for Flash Programming Tips for LPC5500 Series.

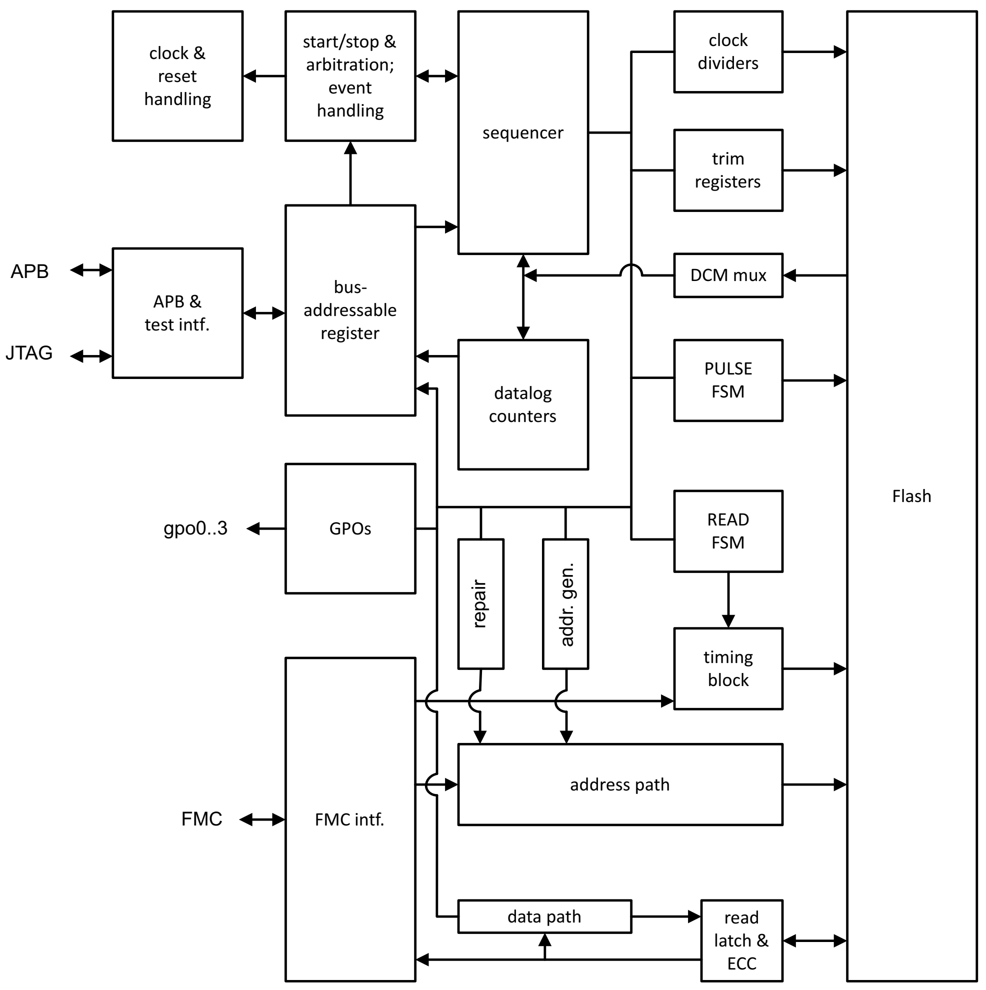


Fig 7. Block diagram

5.4 Software Interface

See Chapter 9 "LPC55S0x/LPC550x 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 168 for a list of registers. Registers are arranged in an address space which is 4k byte wide.

The “Access” field must be interpreted as follows: R = read, W = write, S = set (sets asserted bits, leaves others unchanged), C = clear, (clears asserted bits, leaves 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 explicitly specified, read access is not supported.

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

Table 168. Register overview: flash (base address = 0x40034000) bit description

Name	Access	Offset	Description	Reset value	Section
CMD	W	0x0	Command register.	0x0	5.6.1.1
EVENT	W	0x4	Event register.	undefined	5.6.2.4
STARTA	RW	0x10	Start (or only) address for next flash command.	undefined	5.6.2.1
STOPA	RW	0x14	End address for next flash command, if command operates on address ranges.	undefined	5.6.2.1
DATAW0	RW	0x80	Data register, word 0-7, Memory data, or command parameter, or command result.	0x0	5.6.2.3
DATAW1	RW	0x84	Data register, word 0-7, Memory data, or command parameter, or command result.	0x0	5.6.2.3
DATAW2	RW	0x88	Data register, word 0-7, Memory data, or command parameter, or command result.	0x0	5.6.2.3
DATAW3	RW	0x8C	Data register, word 0-7, Memory data, or command parameter, or command result.	0x0	5.6.2.3
INT_CLR_ENABLE	W	0xFD8	Clear interrupt enable bits.	undefined	5.6.3.1
INT_SET_ENABLE	W	0xFDC	Set interrupt enable bits.	undefined	5.6.3.2
INT_STATUS	R	0xFE0	Interrupt status bits.	undefined	5.6.3.3
INT_ENABLE	R	0xFE4	Interrupt enable bits.	undefined	5.6.3.4
INT_CLR_STATUS	W	0xFE8	Clear interrupt status bits.	undefined	5.6.3.5
INT_SET_STATUS	W	0xFEC	Set interrupt status bits.	undefined	5.6.3.6
MODULE_ID	R	0xFFC	Controller +Memory module identification.	0xC40A0B00	5.6.3.7

5.6 Register description

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

A more detailed description for some bit fields can be found in Section 5.7 “Functional description” where 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 generally 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 stall.

Remark: A command is pending if the initiating bus transaction has already occurred, however, the sequencer waits for the completion of an ongoing read operation before starting.

5.6.1.1 Command register

The controller manages the execution of *commands*. The *commands* encompass any action performed by the controller, for example, a mode change, or programming, erasing, or calculating a checksum over an address range. See Section 5.6.4 “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, and a mode specification. Parameters must be written into corresponding registers before the command is started. The writing of parameters has no effect until the command is started.

Command execution is triggered when writing to the CMD register.

When a command is executed, it sets the 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.

Table 169. Command register (CMD, offset = 0x0) bit description

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

5.6.2 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 in 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

5.6.2.1 Start address register

Table 170. Start (or only) address for next flash command (STARTA, offset = 0x10) bit description

Bit	Symbol	Value	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

5.6.2.2 Stop address register

Table 171. End address for next flash command, if command operates on address ranges (STOPA, offset = 0x14) bit description

Bit	Symbol	Value	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

5.6.2.3 Data register

Table 172. Data register, word 0-3, Memory data, or command parameter, or command result. (DATAW0-3, offset = 0x80 to 0x08C bit description

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

5.6.2.4 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 also be generated 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, or a wake-up from a power-down.

The event register generates 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 173. Event register (EVENT, offset = 0x4) bit description

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

5.6.3 Interrupt and Identification registers

5.6.3.1 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 174. Clear interrupt enable bits (INT_CLR_ENABLE, offset = 0xFD8) bit description

Bit	Symbol	Value	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

5.6.3.2 Set interrupt enable bits register

Table 175. Set interrupt enable bits (INT_SET_ENABLE, offset = 0xFDC) bit description

Bit	Symbol	Value	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

5.6.3.3 Interrupt status bits register

Table 176. Interrupt status bits (INT_STATUS, offset = 0xFE0) bit description

Bit	Symbol	Value	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

5.6.3.4 Interrupt enable bits

Table 177. Interrupt enable bits (INT_ENABLE, offset = 0xFE4) bit description

Bit	Symbol	Value	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

5.6.3.5 Clear interrupt status bits

Table 178. Clear interrupt status bits (INT_CLR_STATUS, offset = 0xFE8) bit description

Bit	Symbol	Value	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

5.6.3.6 Set interrupt status bits

Table 179. Set interrupt status bits (INT_SET_STATUS, offset = 0xFEC) bit description

Bit	Symbol	Value	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

5.6.3.7 Identification register

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

Table 180. Controller and Memory module identification (MODULE_ID, offset = 0xFFC) bit description

Bit	Symbol	Value	Description	Reset value
7:0	APERTURE		Aperture i.	0x0
11:8	MINOR_REV		Minor revision i.	0xB
15:12	MAJOR_REV		Major revision i.	0x0
31:16	ID		Identifier.	0xC40A

5.6.4 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 the 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 in the case where no command could be executed), no command result is defined, not even a fail status is generated. 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 remains undefined after a command execution is attempted.

When a register (STARTA, STOPA, DATAWx, etc.,) contains 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 five 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 181. 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 182)		When this command is initiated, the flash and controller enter power-down mode. During power-down (as with any other command), the flash is not accessible and 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 182 for the meaning of each bit)	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 depends on 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: the exact value is 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 183.	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 181. 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), in which case 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 181. 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 182.

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

Table 183.

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 5.6.4 “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 168), 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 168, 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 168 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 5.6.4 “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 Code 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 arrives.

```
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 period of time (all needed code is in ROM/RAM), and so temporarily turns it off. 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, the INT_STATUS register handling can be removed except when the flash is accessed after a wake-up is triggered and before the flash is ready, in which case 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 5.6.4 “Command listing (CMD)”.

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

5.7.8 Writing

A number of APB writes are required to fully define a memory word that 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 transfers 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 various 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 so 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 by writing to INT_SET_STATUS, in order to test the application's behavior, 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 if the application urgently requires access to the memory, such as in the case where there are erase and program commands which take a long time to complete.

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 must be retried, even if the memory content appears to be intact (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 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).

A simple example is provided below that shows how a small amount of regularly modified data can be handled to guarantee that, in case of power loss during a modification, valid data can always be 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 available (for example, not modified by a hacker, a programming error, and a 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). Note that the "||" symbol indicates a concatenation of data.

```
//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, its condition and location 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 a subsequent error in the same word from causing 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 a successful erase.
- Due to the presence of ECC, over-programming an already programmed word will likely result in inconsistent parity bits. For this reason, programming a memory word, without first erasing it, is not allowed.
- If a program or erase operation is aborted, data and parity bits are unknown and probably inconsistent. In this case, the operation may result in unpredictable behavior and results (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, 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 other errors (correctable or uncorrectable) are separately counted (a saturating counter is used). A controller command allows this information to be read and contextually clears 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 CPU (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 CPU is set high for at least one clock cycle.

The above provision is to ensure that no event is lost in the event that 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.

To maintain system integrity, the interrupt request to the CPU 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).

UM11424

Chapter 6: LPC55S0x/LPC550x Boot ROM

Rev. 1.5 — 21 December 2023 **User manual**

6.1 How to read this chapter

This chapter applies to all LPC55S0x/LPC550x 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 9 “LPC55S0x/LPC550x Flash API”), supports automated booting from internal flash.
- IAP calls. See Chapter 8 “LPC55S0x/LPC550x ISP and IAP”.
- FLASH API for programming internal flash. See Chapter 5 “LPC55S0x/LPC550x Flash”.
- Supports SPI flash recovery boot from 1-bit SPI flash device. 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 LPC55S0x/LPC550x 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 LPC55S0x/LPC550x will read status of the ISP pins to determine boot source. See Table 184.

Table 184. Boot mode and ISP download modes based on ISP pins

Boot mode	ISP0 (PIO0_5 pin)	Description
Passive boot	HIGH	The LPC55S0x/LPC550x will look for valid image in the internal flash, if no valid image is found, the LPC55S0x/LPC550x will enter ISP boot mode based on DEFAULT_ISP_MODE bits defined in Table 185.
ISP boot	LOW	One of the serial interfaces (UART0, I^{2}C1, SPI3, HS_SPI) is used to download image from host into internal flash. The first valid probe message on USART, I^{2}C or SPI locks in that interface.

Table 185. 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 active peripheral is probed from one of following serial interfaces, and the image is downloaded from the probed peripherals: UART0, I^{2}C1, SPI3, HS_SPI.
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.
I^{2}C Slave ISP	1	0	0	The I^{2}C slave is used to download the image.
Disable ISP	1	1	1	Disable ISP mode.

Table 186 shows the ISP pin assignments and is the default pin assignment used by the ROM code that cannot be changed.

Table 186. ISP pin assignments

ISP pin	Port pin assignment
ISP0	PIO0_5
USART ISP mode	
FC0_TXD	PIO0_30
FC0_RXD	PIO0_29
I^{2}C ISP mode	
FC1_SDA	PIO0_13
FC1_SCL	PIO0_14
SPI Flash Recovery mode	
FC3_TXD_SCL_MISO_WS	PIO0_2
FC3_RXD_SDA_MOSI_DATA	PIO0_3
FC3_CTS_SDA_SSELN0	PIO0_4
FC3_SCK	PIO0_6
SPI ISP mode	
HS_SPI_SCK	PIO1_2
HS_SPI_SSEL1	PIO1_1
HS_SPI_MISO	PIO1_3
HS_SPI_MOSI	PIO0_26

Figure 8 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 187. 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 187. 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 187. Image header for the LPC55S0x/LPC550x 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.

Table 187. Image header for the LPC55S0x/LPC550x device

Offset	Size in bytes	Symbol	Description
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

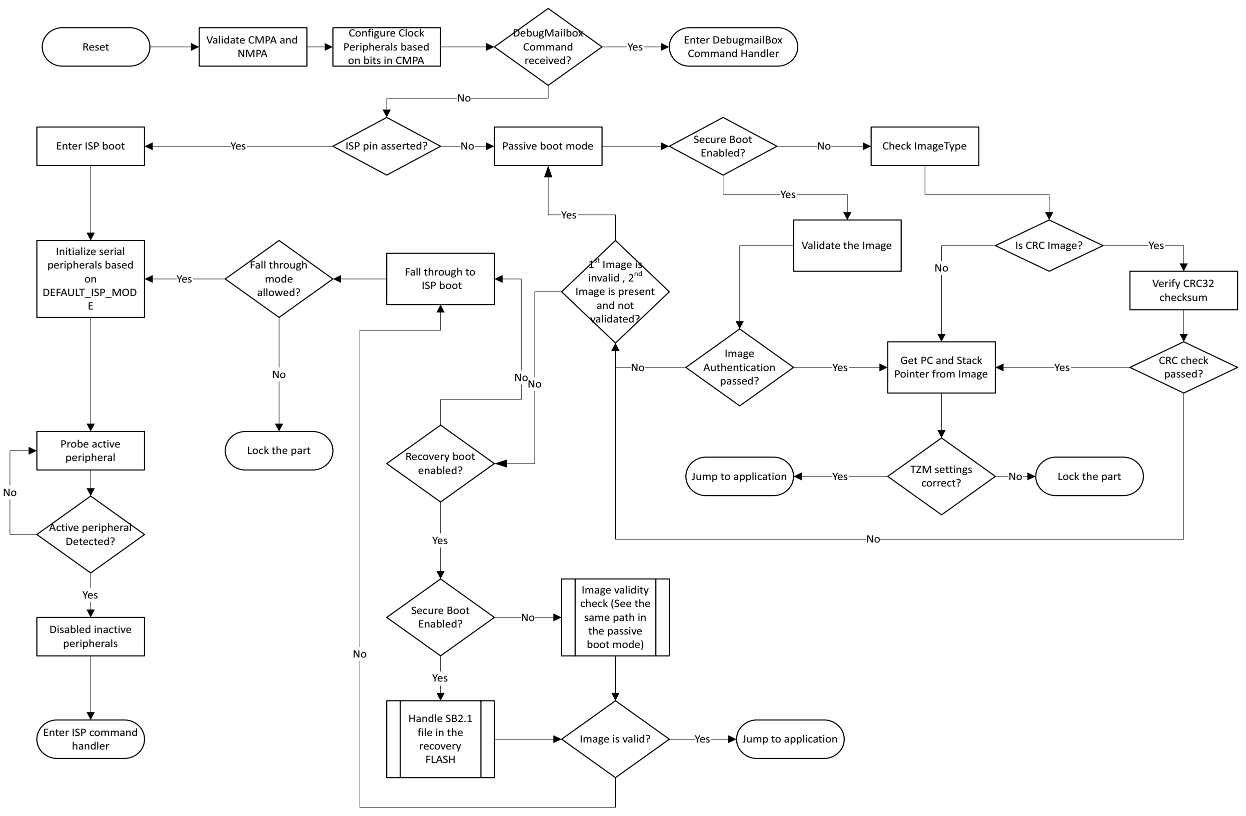

Reset
Validate CMPA and NMPA
Configure Clock Peripherals based on bits in CMPA
DebugMailbox Command received?
Yes
Enter DebugmailBox Command Handler
No
Enter ISP boot
Yes
ISP pin asserted?
No
Passive boot mode
Secure Boot Enabled?
No
Check ImageType
Yes
Validate the Image
Is CRC Image?
Yes
No
Verify CRC32 checksum
Initialize serial peripherals based on DEFAULT_ISP_MODE
Yes
Fall through mode allowed?
Fall through to ISP boot
1st Image is invalid , 2nd Image is present and not validated?
No
Image Authentication passed?
Yes
Get PC and Stack Pointer from Image
Yes
CRC check passed?
No
Lock the part
No
No
Recovery boot enabled?
Jump to application
Yes
TZM settings correct?
No
Lock the part
Probe active peripheral
No
Active peripheral Detected?
Yes
Disabled inactive peripherals
Enter ISP command handler
Yes
Secure Boot Enabled?
No
Image validity check (See the same path in the passive boot mode)
No
Yes
Handle SB2.1 file in the recovery FLASH
Image is valid?
Yes
Jump to application

Fig 8. LPC55S0x/LPC550x boot flow chart

6.4 Boot modes

The boot modes include:

- Section 6.4.1 boot mode”.
- Section 6.4.2 “ISP boot mode”
- Section 6.4.3 “SPI flash recovery”

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 8.

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 “LPC55S0x/LPC550x ISP and IAP”.

6.4.3 SPI flash recovery

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 7.3.3.3 “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 (0x3E404) 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 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 0x3DE00. Please see: Chapter 9 "LPC55S0x/LPC550x Flash API") Protected Flash Region for details.

Table 188. Image header for the LPC55S0x/LPC550x devices

Region	Field	Description
0x3DE00-0x3E3FF	Customer In-field Programmable Area(CFPA)	See the Secure Boot chapter for more details.
0x3E400-0x3EBFF	Customer Manufacturing/Factory Programmable Area (CMPA) and Key Store Area (KSA)	
0x3EC00-0x3FDFF	NXP Manufacturing Programmed Area(NMPA)	Reserved for NXP internal use only

UM11424

Chapter 7: LPC55S0x Secure Boot ROM

Rev. 1.5 — 21 December 2023 **User manual**

7.1 How to read this chapter

This document describes the Secure Boot ROM architecture for the LPC55S0x series.

The Secure part of ROM boot loader provides the 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 prevents unauthorized code from being executed on a given product. It achieves this level of security by always leaving the device's ROM in an executing mode when coming out of a reset. This allows the ROM to 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, through the verification of digital signatures associated with the boot code.

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. Support is provided for up to four Root of Trust keys.

The device can be configured to boot plain images during development. In this case, the ROM does not check the image to be booted, or performs only CRC32 checking, depending on the 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 updates simply perform 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. The SB file can be loaded via secure interfaces such as UART, etc., or can be provided to the ROM API as a complete binary file. See: Section 6.4.3 "SPI flash recovery" for more 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 when 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 specifications:

- DICE (Device Identifier Composition Engine)
- Load of TrustZone-M pre-configuration during ROM secure boot
- Booting from encrypted internal Flash regions using PRINCE peripheral module
- Debug authentication
- Initial boot state, as specified in ARM Platform Security Architecture Security Model 1.0
- Device provisioning (ROM embedded support for the initial secure provisioning of the boot keys)

7.2.5 Boot flow diagram

Booting of the device is controlled by a setting written in the PFR (Protected Flash Region) of internal device flash memory and based on ISP pin settings as shown in Figure 9 "Secure Boot ROM Flow chart".

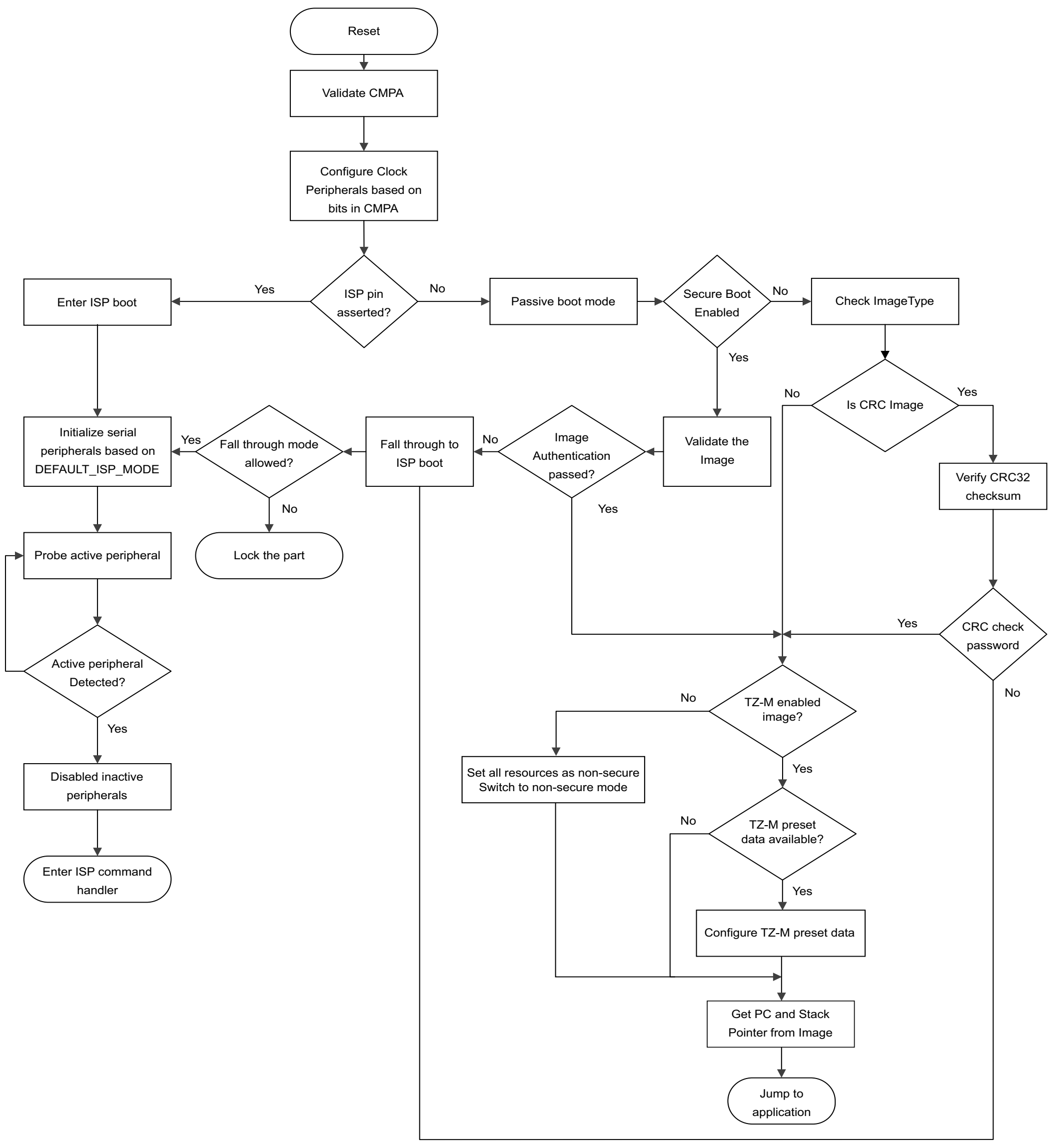

Reset
Validate CMPA
Configure Clock Peripherals based on bits in CMPA
ISP pin asserted?
Yes
No
Enter ISP boot
Passive boot mode
Secure Boot Enabled
No
Yes
Check ImageType
Is CRC Image
No
Yes
Verify CRC32 checksum
Validate the Image
Image Authentication passed?
No
Yes
Fall through to ISP boot
Fall through mode allowed?
Yes
No
Lock the part
Initialize serial peripherals based on DEFAULT_ISP_MODE
Probe active peripheral
Active peripheral Detected?
Yes
Disabled inactive peripherals
Enter ISP command handler
CRC check password
Yes
No
TZ-M enabled image?
No
Yes
Set all resources as non-secure Switch to non-secure mode
TZ-M preset data available?
No
Yes
Configure TZ-M preset data
Get PC and Stack Pointer from Image
Jump to application

Fig 9. Secure Boot ROM Flow chart

7.2.6 Data structures

7.2.6.1 Overview

LPC55S0x stores configuration and PUF key store for the boot ROM in Protected Flash Region (PFR). It resides at the end of the flash region and can be programmed through ROM in ISP mode.

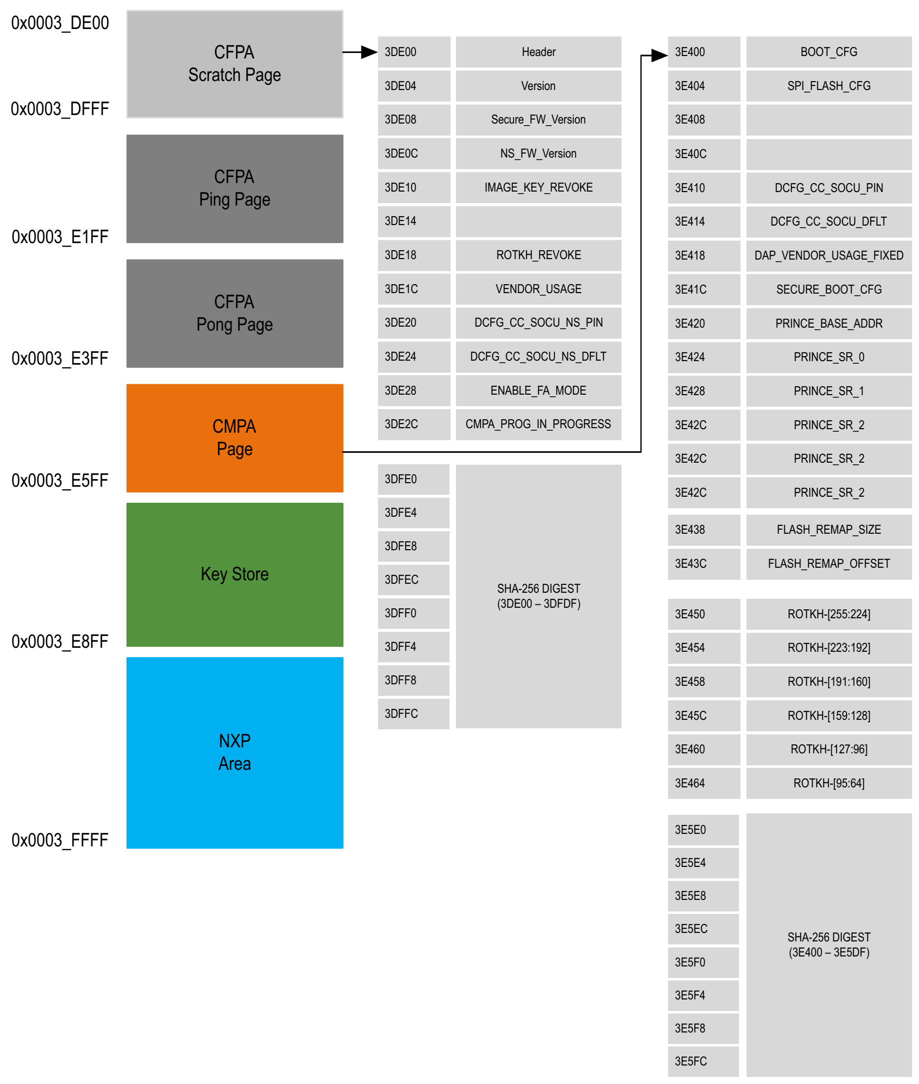


Fig 10. Protected Flash Region

7.2.6.2 Key storage in Protected Flash Region

LPC55S0x uses PUF controller for key wrapping. The PUF key store occupies three flash pages (1536 bytes in total) of PFR and consists of an Activation Code and six Key Codes which are managed and used mainly by the ROM during the boot and SB file processing. The key store data structure can be created during the key provisioning process and written to PFR using the *write to non-volatile* blhost command. The content of key storage is also available to a user application by using the **PFR_KeystoreGetAC** and **PFR_KeystoreGetKC** ROM API functions. During the startup, the ROM checks if a valid key store data structure is present in PFR. If so, the whole key store data structure is loaded into RAM and ROM issues the PUF start procedure, which initializes PUF and loads the activation code so that each key can be used as needed.

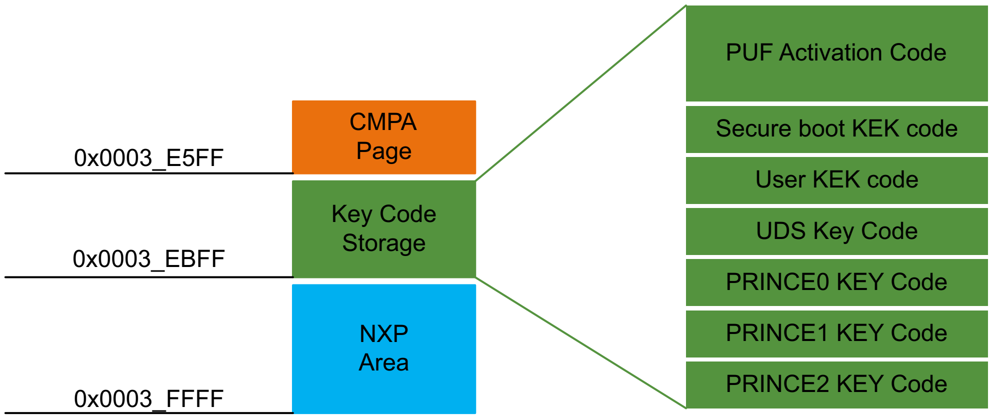


Fig 11. KeyStore area in PFR

Table 189. PUF key code storage area structure

Address	Size (bytes)	Name	Description
0x3E600	4	Key Store Header	Marker. A value of 0x95959595 means that Activation code is valid.
0x3E604	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.
0x3E608	1192	Activation Code	Device specific PUF activation code generated by enroll command during key provisioning.
0x3EAB0	56	SBKEK Key Code	Key Code for wrapped SBKEK key.
0x3EAE8	56	USERKEK Key Code	Key Code for wrapped USERKEK key.
0x3EB20	56	UDS Key Code	Key Code for wrapped UDS key.
0x3EB58	56	PRINCE Region 0 Key Code	Key Code for wrapped PRINCE Region 0 key.
0x3EB90	56	PRINCE Region 1 Key Code	Key Code for wrapped PRINCE Region 1 key.
0x3EBC8	56	PRINCE Region 2 Key Code	Key Code for wrapped PRINCE Region 2 key.

7.3 Keys

Key Codes for given keys are generated (wrapped) using PUF by data supplied to blhost by key provisioning commands along with the specified key type and key length. Keys stored with PUF key index 0 can be unwrapped only on a secure key hardware bus so that only security peripherals connected to this bus are able to use this key. Depending on the value of "Enable HW user mode keys", specified by the bit in Image Type (word at offset 0x24), access to this secure key bus form a non-secure method can be restricted. When this bit is set to 0, the keys will be accessible by PUF only from a secure environment. See Chapter 44 "LPC55S0x 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 190. PUF key code format

Offsets	Bytes	Name	Description
0x0	4	Key code marker	A value of 0x59595959 means that the Key code is valid.
0x4	52	Key Code	Key code. Wrapped plaintext key. SRAM PUF device unique key is used as a key wrapping key.

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

Note: PUF key index 0 indicates that the key unwraps to a dedicated hardware bus, directly connected to the AES cryptographic engine.

USERKEK – User Key Encryption Key

- Not used by LPC55S0x bootloader. Available for user as pre-shared master key
- AES-256 symmetric key
- blhost key type 11
- PUF key index 0

UDS key – Unique Device Secret

- Unique Device Secret for DICE
- HMAC-SHA256 256-bit symmetric key
- blhost key type 12
- PUF key index 15

Note: PUF key index 15 indicates that the key unwraps to system memory. This index is only available during ROM execution. When ROM exits to a user application or enters debug mode, PUF key index 15 is locked by the ROM.

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 a 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 191. configuration overview

Address	Bytes	Name	Description
0x3E41C	4	SECURE_BOOT_CFG	Secure boot configuration flags.
0x3E420	4	PRINCE_BASE_ADDR	PRINCE configuration and region base addresses.
0x3E424	4	PRINCE_SR_0	Region 0, sub-region enable.
0x3E428	4	PRINCE_SR_1	Region 1, sub-region enable
0x3E42C	4	PRINCE_SR_2	Region 2, sub-region enable
0x3E450	32	RKTH	Root Key Table Hash.

SECURE_BOOT_CFG configuration word

Table 192. SECURE_BOOT_CFG word bit field definitions

Address	Bit(s)	Name	Description
0x3E41C	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

Table 192. **SECURE_BOOT_CFG word bit field definitions**

Address	Bit(s)	Name	Description
	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. See security epoch PFR fields below this table. 2'b00: not included 2'b01: included 2'b10: included 2'b11: included
	17:16	SKIP_BOOT_SEED	Skip boot seed computation 00: enable BOOT_SEED 01,10,11: disable BOOT_SEED
	19:18	BOOT_SEED_INC_NXP_CFG	Include NXP area in BOOT SEED computation 00: not included 01, 10, 11: included
	21:20	BOOT_SEED_CUST_CFG	Include CMPA area in BOOT SEED computation 00: not included 01, 10, 11: included
	23:22	BOOT_SEED_INC_EPOCH	Include security epoch area in BOOT_SEED computation. See security epoch PFR fields below this table.
	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)

Security epoch consists of the following PFR fields:

IMAGE_KEY_REVOKE

ROTKH_REVOKE

VENDOR_USAGE

DCF_CC_SOCU_NS_PIN

DCF_CC_SOCU_NS_DFLT

PRINCE_BASE_ADDR: Contains various configuration for PRINCE peripherals to be set by ROM bootloader during device startup.

Table 193. PRINCE configuration

Address	Bit(s)	Name	Description
0x3E420	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 region0 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 region1 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 region0 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 region1 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 region2 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	

PRINCE_SR_x: When on-the-fly encryption/decryption of internal flash using PRINCE is enabled, ROM configures the sub-region enable bits for a given memory region according to a value stored in this word as shown in Table 194 "PRINCE sub-region enable bits".

Table 194. PRINCE sub-region enable bits

Address	Bit(s)	Name	Description
0x3E424	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 a 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 in Figure 12 "RKTH generation process".

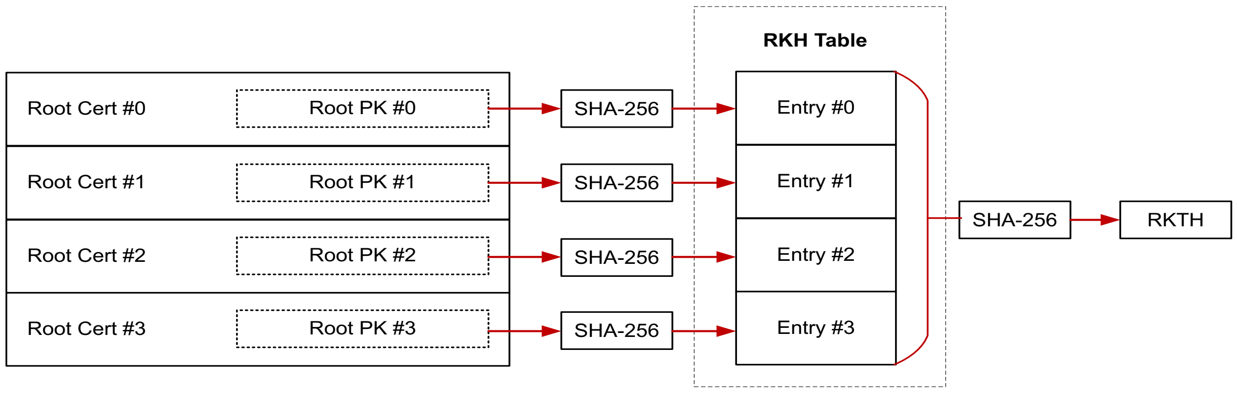


Fig 12. RKTH generation process

Each entry in the table is an SHA-256 computed over the concatenation of an RSA public key's modulus and exponent (modulus || exponent, where the "||" symbol indicates a concatenation of data.). 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 range 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 are held in reserve in case a public key has to be revoked. The customer is responsible for implementing the mechanism to determine whether a key needs to be revoked, and to then set the appropriate `RKTH_REVOKE` bit(s). This is usually accomplished 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 195. RKTH layout in CMPA

Address	Description	Address	Description
0x3E450	RKTH[255:224]	0x3E460	RKTH [127:96]
0x3E454	RKTH [223:192]	0x3E464	RKTH [95:64]
0x3E458	RKTH [191:160]	0x3E468	RKTH [63:32]
0x3E45C	RKTH [159:128]	0x3E46C	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 196. CFPA page layout

Address	Byte(s)	Name	Description
0x3DE08	4	Secure_FW_Version	Secure firmware version (Monotonic counter)
0x3DE0C	4	NS_FW_Version	Non Secure firmware version (Monotonic counter)
0x3DE10	4	IMAGE_KEY_REVOKE	Image key revocation ID
0x3DE18	1	RKTH_REVOKE	Used for revocation of individual Root keys
0x3DE30	56	PRINCE Region 0 IV CODE	IV code used for PRINCE region 0
0x3DE68	56	PRINCE Region 1 IV CODE	IV code used for PRINCE region 1
0x3DEA0	56	PRINCE Region 2 IV CODE	IV code used for PRINCE region 2

Secure_FW_version: Optionally used during SB2 file loading. The value written in the configuration word must always be lower or equal to the secure FW version specified in the elftosb.bd file used to create the SB2 file. Otherwise, if this version check command is included in the SB2 file, the file load will be rejected.

NS_FW_version: Optionally used during SB2 file loading. The value written in the configuration word must always be lower or equal to the non-secure FW version specified in the elftosb.bd file used to create the SB2 file. Otherwise, if this version check command is included in the SB2 file, the file load is rejected.

IMAGE_KEY_REVOKE: This value is checked during the 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 fails. 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 damaging the device if power loss happens after a 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 a revoked RoT key, they will be rejected during the authentication process and fail to boot if SEC_BOOT_EN is set to boot only signed images.

Table 197. RKTH table bit field description

Address	Bit(s)	Name	Description
0x3DE18	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 LPC55S0x as well as Secure versions during development life-cycle state of S parts (LPC55S0x).

The structure of unsigned CRC images is shown in Figure 13 “Structure of Unsigned CRC images”.

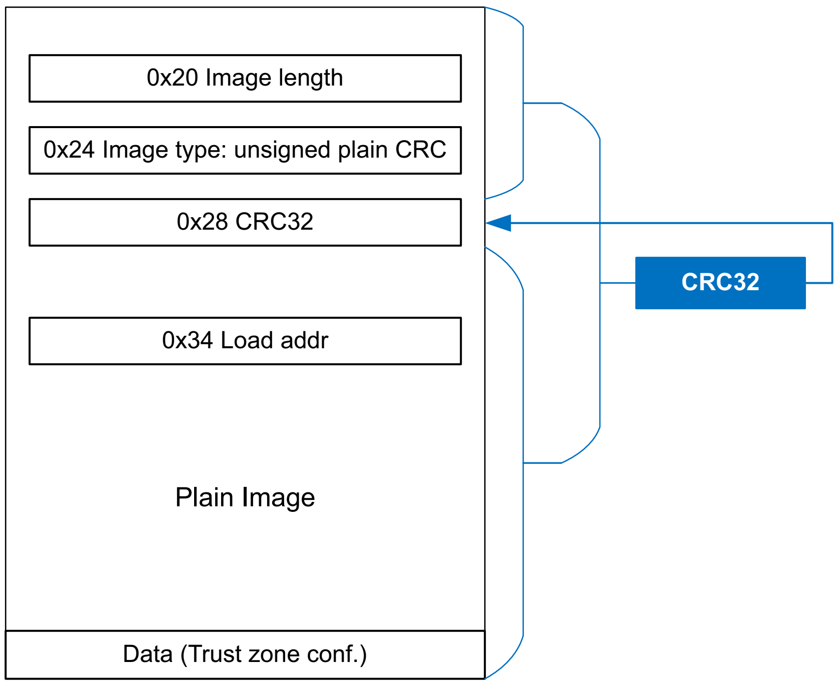


Fig 13. Structure of Unsigned CRC images

Note: When Image Type is 0x0, CRC32 checking is bypassed. Such an image can be used as a generic image during development.

7.3.3.1 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 in Figure 14 "Structure of Signed Images".

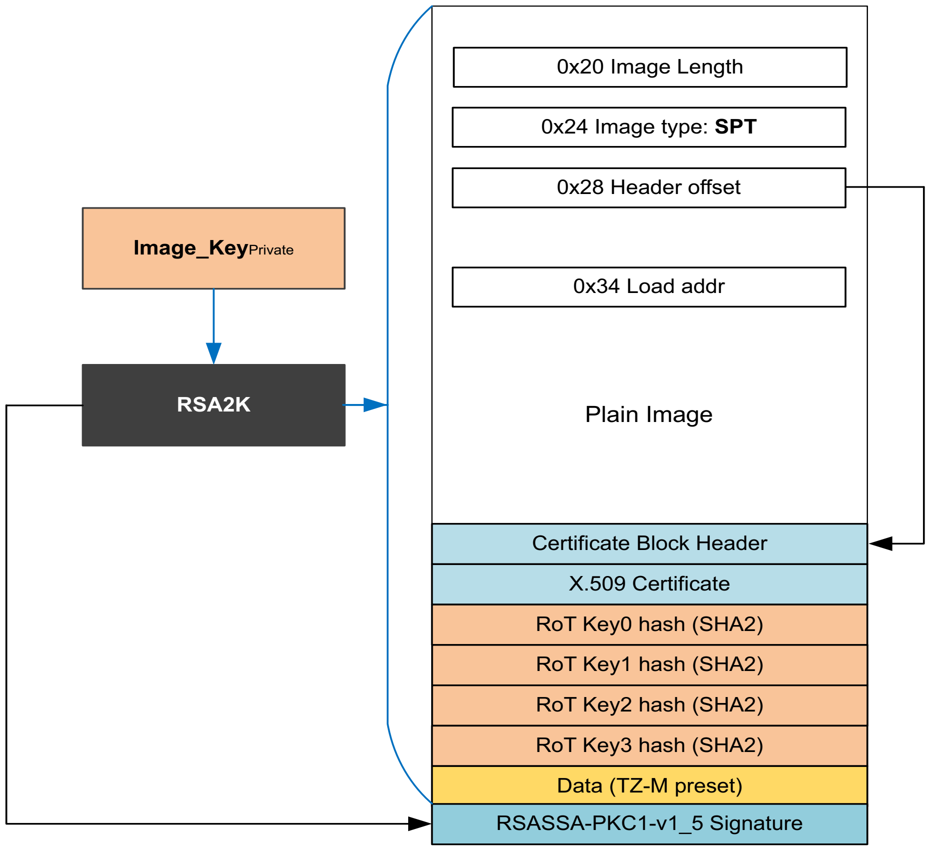


Fig 14. Structure of Signed Images

Image length - total length of the image in bytes including signature

Image type - SPT (Signed Plain Text) = 0x4 or 0x5

Table 198. LPC55S0x 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
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	Enable HW user mode keys	0: HW bus keys are available to Secure world only 1: HW bus keys are available to Secure and Non-secure world
11:8	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 199.

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

7.3.3.2 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 is shown in Figure 15 “Structure of Certificate Block”.

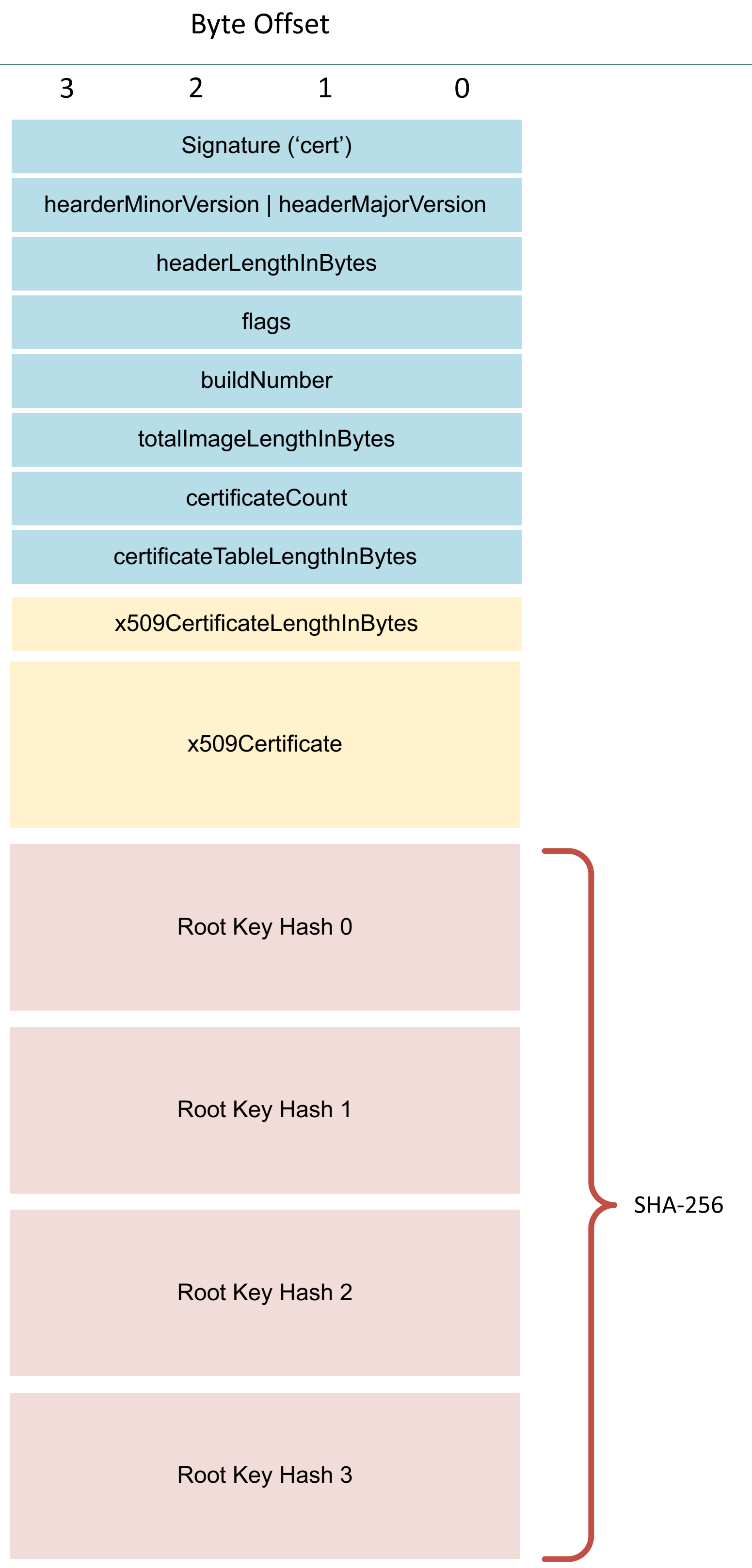


Fig 15. Structure of Certificate Block

7.3.3.2.1 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 200.

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 minBuildNumberspecified 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)).

7.3.3.2.2 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 and they 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.

7.3.3.3 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 SB2 file can be considered as a type of script (commands and data), for which the ROM is the interpreter. Version 2.1 updates the encryption scheme to use modern algorithms, adds support for signed images and it also makes the usage of a digital signature mandatory. The LPC55S0x silicon supports version 2.1 of the SB image format. The ReceiveSBFile command verifies the digital signature.

The SB 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 2.1 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. This chapter provides an introductory description of the SB file format components, while it is left to the elftosb Tool User’s Guide to provide additional details.

7.3.3.3.1 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.

7.3.3.3.2 MAC of the section MAC table

The expected MAC of the Section MAC table. During image verification, the actual MAC of the Section MAC table is computed and compared with the expected MAC. Note that the expected MAC value itself is provided by the digital signature verification.

7.3.3.3.3 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 avaiable) 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.

7.3.3.3.4 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.

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.

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.

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 support for the following bootloader commands:

WriteMemory, FillMemory, ConfigureMemory, FlashEraseAll, FlashEraseRegion, SecureFirmwareVersionCheck, NonsecureFirmwareVersionCheck

The WriteMemory and FillMemory commands can be used to write data to RAM. WriteMemory can be also used to program internal flash, including the PFR CFPA page, assuming the flash is first erased, for example, by the FlashEraseAll or FlashEraseRegion commands. The ConfigureMemory command can be used to configure the LPC55xx PRINCE on-the-fly encryption module.

The recovery boot mode using the SB 2.1 file supports only four commands as follows:

- ROM_NOP_CMD
- ROM_LOAD_CMD
- ROM_JUMP_CMD

- ROM_FW_VER_CHK_CMD
- ROM_TAG_CMD
- ROM_FILL_CMD
- ROM_ERASE_CMD
- ROM_MEM_ENABLE_CMD
- ROM_PROG_CMD

Data section: The loader considers any section with the bootable section flag cleared as data and does not examine the contents of such sections; it simply skips over them. Data sections may optionally be unencrypted by setting the cleartext flag.

Certificate block header, certificates and RKH table: For SB 2.1, the certificate block header, certificate chain and RKH table are mandatory components of the SB 2.1 file header.

Signature: For SB 2.1, the signature is mandatory and immediately follows the RKH table.

The SB 2.1 file header has the same structure as a signed image in the internal flash. Thus, the ROM's image authenticate function is used to verify digital signatures of internal flash images for the SB 2.1 header. The signature in RSASSA-PKCS1-v1_5 format is appended to the tail end of the internal flash image and to the tail end of the SB 2.1 header.

7.3.3.3.5 Usage of firmware update

SB 2.1 files are always encrypted, and the header is always signed. The loader API is called from the application code to authenticate an image, either signed code or an SB file.

The recommended method for performing secure firmware updates is as follows:

- A user application receives an encrypted SB file containing new firmware and stores it in external SPI flash, or a similar memory store.
- The API is used to authenticate the SB file.
- The API is then used 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.

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 setup with the 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 UDS; UDS key type = 12.

blhost -p com6 -- key-provisioning set_key 12 32

:: 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

The SB 2.1 processing by ROM also depends on the presence of RKT hash (RKTH) in the CMPA page in the PFR.

7.3.3.3.6 Secure ROM API

The ROM API table is located at address *0x1301fe00* and contains absolute ROM API function addresses which can be called using function pointers. Th PRINCE ROM API section starts at address *0x1300507c* and *skboot_authenticate()* function address is located at *0x130050ec*. Only secure boot related functions are described in this chapter.

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 the PRINCE on-the-fly encryption/decryption peripheral.

Table 201. Secure ROM API summary

Address in ROM API table	Absolute function address	Function
0x130050ec	0x1300c3ff	skboot_status_t skboot_authenticate(const uint8_t *imageStartAddr, secure_bool_t *isSignVerified)
0x130050f0	0x13007883	void skboot_hashcrypt_irq_handler(void)
0x1300507c	0x1300a873	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)
0x13005080	0x1300a999	skboot_status_t bus_crypto_engine_load_iv(uint32_t region, uint8_t *iv_code)
0x13005084	0x1300a695	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 the 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 2.1 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 Verification.

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 LPC55S0x device is:

0x30000000 to 0x30007FFF

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 passes.

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_TRACKER_VERIFIED (0x55aacc33U) -- when signature verification passes.

*isSignVerified = kSECURE_FALSE (0x5aa55aa5U) -- when signature verification fails.

7.4.2 HASH_IRQHandler

This function shall be called from the 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 0x1301fe00 and the PRINCE ROM API part locates at address 0x1300507c.

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 in 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 202. 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)0x1301fe00)

#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 203. 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)0x1301fe00)

#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]);

7.5.3 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 "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 204. 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)0x1301fe00)

#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);

7.5.3.1 ROM TrustZone support

7.5.3.1.1 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 205. Trustzone image type

TZM_IMAGE_TYPE value (offset 24, bit 14)	
0	TrustZone enabled image
1	TrustZone disabled image

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 LPC55S0x 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 perform any software modification.

Note: After a 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. Code execution or data read from this memory otherwise should be avoided, otherwise a HardFault is generated.

TrustZone enabled image: A TrustZone enabled image that is executed on devices with TrustZone (M33 with security extension). In this case, the user application is split into secure and non-secure partitions, and after a device reset, the software execution starts in secure mode. If the TrustZone enabled image is executed, the ROM doesn't provide any TrustZone settings (except optional TrustZone preset data configuration) and jumps into the user application in secure mode. The executed software/image is responsible for TrustZone settings and jump from secure to normal modes.

7.5.3.1.2 TrustZone preset data

Support provides support for TrustZone data configuration during the boot process. The TrustZone preset data includes:

- VTOR, VTOR_NS, NVIC_ITNS0, NVIC_ITNS1 (CPU0) registers
- 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 the lock feature and a corresponding bit is set, the register is also locked, so any further register modification is not possible until the next reset.

This feature increases the robustness of the user application as it jumps into a pre-configured TrustZone environment and it doesn't need to contain any TrustZone configuration code.

TrustZone preset data structure: The TrustZone preset data structure is defined by the 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 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_rule;   /*!< RAMX Memory Rule Register 0 */
uint32_t ram0_slave_rule; /*!< RAM0 Slave Rule Register */
uint32_t ram0_mem_rule;   /*!< RAM0 Memory Rule Register 0 */
uint32_t ram1_slave_rule; /*!< RAM1 Slave Rule Register */
uint32_t ram1_mem_rule;   /*!< RAM1 Memory Rule Register 0 */
uint32_t ram2_slave_rule; /*!< RAM2 Slave Rule Register */
uint32_t ram2_mem_rule;   /*!< RAM2 Memory Rule Register 0 */
uint32_t ram3_slave_rule; /*!< RAM3 Slave Rule Register */
uint32_t ram3_mem_rule;   /*!< RAM3 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_rule;    /*!< AHB Peripherals 2 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_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 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 206. 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 Figure 16 “Location of TrustZone configuration data in the image file”.

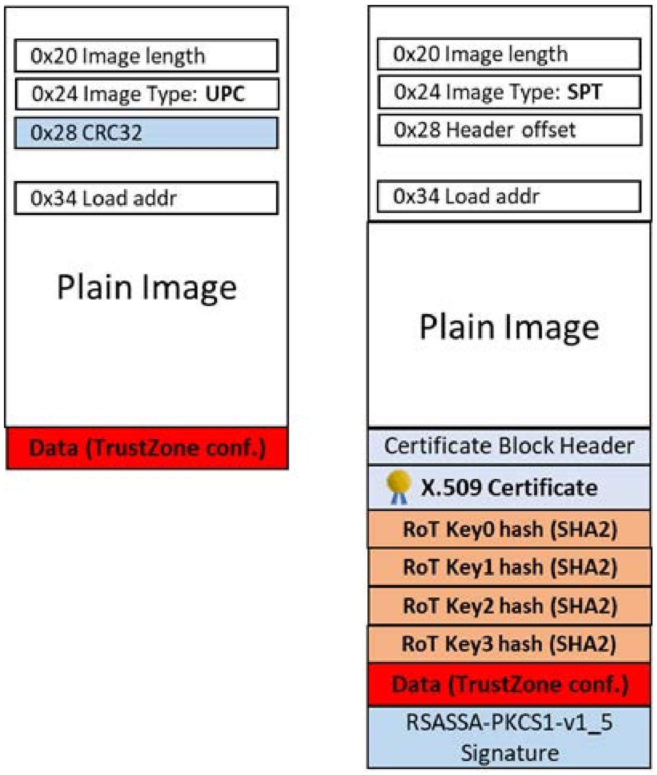


Fig 16. Location of TrustZone configuration data in the image file

The elftosb.exe tool can also be used to create TrustZone configuration data. For more information please see the elftosb manual.

TrustZone configuration data is defined in the vector section of the image header at offset 0x24, bit 13 (TZM_PRESET).

Table 207. 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. The whole ROM space (0x13000000-0x1301FFFF) must be configured as secure privilege.
2. When a secure MPU is used, the whole ROM space (0x13000000-0x1301FFFF) must be configured for code execution.

If these two conditions are not met, the boot process fails.

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 208. 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

7.5.3.1.3 Boot ROM API and TrustZone

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.

TrustZone enabled images: The whole boot ROM is executed in secure mode which allows full control of which ROM API functions are available. 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. This 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)0x1301fe00)
```

```
#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));
```

7.5.3.2 Secure boot usage

The LPC55S0x 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 – see AN12283 LPC55S0x Secure Boot for detailed step-by-step guide describing use of these tools.

7.5.3.2.1 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

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 0x3DE18 must be setup to accept signed images with created certificates.

- Enter ISP boot mode by asserting ISP boot pin
- Prepare CMPA page using elftosb 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 PC tool
  – RKTH_REVOKE field to accept signed images with created certificates
- Write prepared CFPA/CMPA page into flash memory using blhost tool

Signed image preparation: NXP provides the elftosb tool which prepares a signed binary that can be loaded to a target device. The input for the elftosb program is a plain application image in binary format, image signing certificate, associated private key and JSON format configuration structure. For a detailed step-by-step guide, see AN12283 LPC55S0x Secure Boot application note.

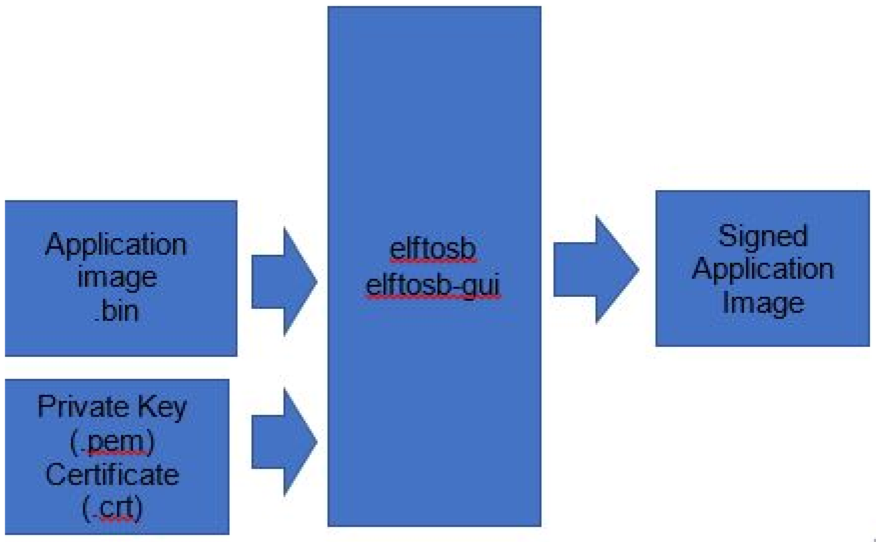


Fig 17. Signed Image Preparation

The following information is needed by the elftosb tool to produce an internal flash (XIP) signed image:

- Plain application binary generated for the LPC55S0x device
- Start address of the application binary
- TZ related settings
- Certificates or chain certificates
- Private key for selected certificate (last certificate in chain)

Loading signed image: The signed image can be programmed directly into the device using various methods:

- 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

7.5.3.2.2 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 the device and is protected against copying using PUF encryption in a format that is readable only for a 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 an internal HW bus. The user application is then decrypted in real time and executed.

The following sections describe the tools required to perform key provisioning and describe how to program the application image using PRINCE.

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 the internal hardware bus.

The following example shows the sequence of commands that must be issued from the PC blhost application to the device in ISP mode in order to generate a proper PRINCE enabled Key Store. The key store is saved into the device PFR and accessed by boot ROM during a secure boot.

In this example, the 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

PRINCE region configuration with blhost: For PRINCE encryption and decryption, the regions and sub-regions for the crypto operation need to be configured. This is accomplished with the ISP “configure-memory” command. This command has to be called with the data structure and ancillary PRINCE commands shown in the following tables and according to the following procedure.

Table 209. Structure for configure-memory command

Offset	Size	Description
0	4	PRINCE Configuration
4	8	PRINCE Region info

Table 210. 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 211. 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 the “configure-memory” command with this structure:

1. Region selection (Region 0 in this example)
   – blhost.exe -p COMxx -- fill-memory 0x20008000 4 0x50000000
2. Start address of encrypted area (Address 0x0 in this example)
   – blhost.exe -p COMxx -- fill-memory 0x20008004 4 0
3. Length of the encrypted area (0x40000 in this example)
   – blhost.exe -p COMxx -- fill-memory 0x20008008 4 0x40000
4. Call configure-memory with prepared structure in RAM
   – blhost.exe -p COMxx -- configure-memory 0 0x20008000

After issuing this command with the appropriate settings, PRINCE is configured for flash encryption.

Note: The PFR area should be excluded from the PRINCE encryption area, i.e., the start and size settings in configuring the structure must be set to avoid overlapping with the PFR area.

Upload image: A "prince erase checker" is implemented in the boot ROM and is designed to check whether the PRINCE sub regions (8k block) are erased at once in a single operation. Similarly, "prince flash write checker" is implemented in the ROM code to check whether the entire range of PRINCE 8 KB subregions are programmed at once. To load the image that is encrypted (on-the-fly) by PRINCE, the following sequence of ISP commands must 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 the device
   – Press reset pin or run BLHOST tool blhost -p COMxx reset

After completing these steps, the image is encrypted and then loaded into flash where it starts executing. Decryption of the flash content is performed on-the-fly by the PRINCE hardware engine.

Note: Special care should be taken when writing and erasing the encrypted flash area. The entire 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 must encompass the entire 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).

Note: Due to limitations in the "prince erase checker" and "prince flash write checker" implementations in the boot ROM, the flash-erase-region and write-memory command parameters should be used to encompass the whole encrypted area. It is recommended that the latest MCUXpressoSDK Prince driver be used for performing the Prince region configuration as well as erase, and write operations.

UM11424

Chapter 8: LPC55S0x/LPC550x ISP and IAP

Rev. 1.5 — 21 December 2023 **User manual**

8.1 How to read this chapter

All LPC55S0x/LPC550x devices include In-System Programming (ISP) functions to support image programming from serial interface (UART, I^2C, SPI). In-Application Programming (IAP) calls are available.

8.2 Features

- In-System Programming supports:
  - Supports UART, I^2C, and SPI 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 “LPC55S0x/LPC550x Boot ROM”

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^{2}C 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 In-Application Programming, see Section 8.11 “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 “LPC55S0x/LPC550x 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^{2}C / SPI/ USART interface. The auto-detect looks for activity on the USART, I^{2}C, and SPI 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^{2}C, and SPI 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).

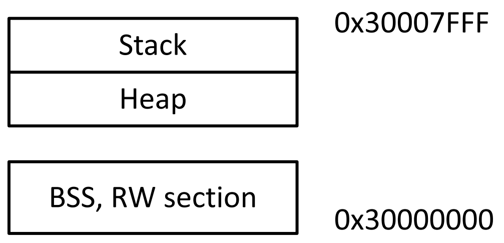


Fig 18. Reserved RAM region for the boot ROM

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)

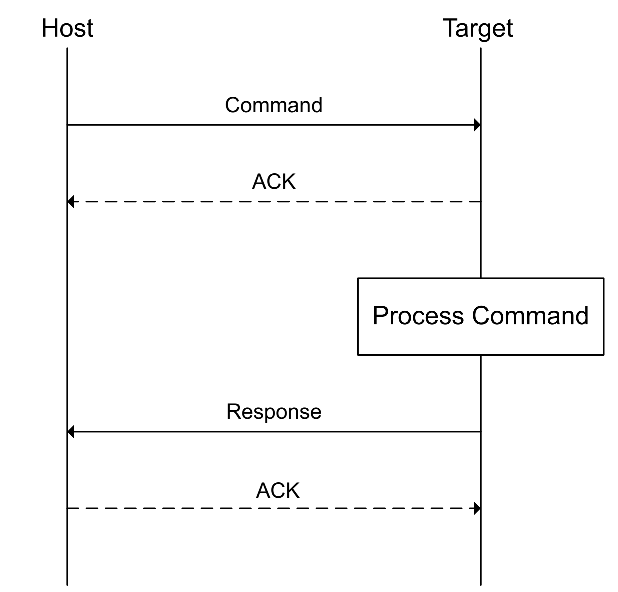


Fig 19. Command with no data phase

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.

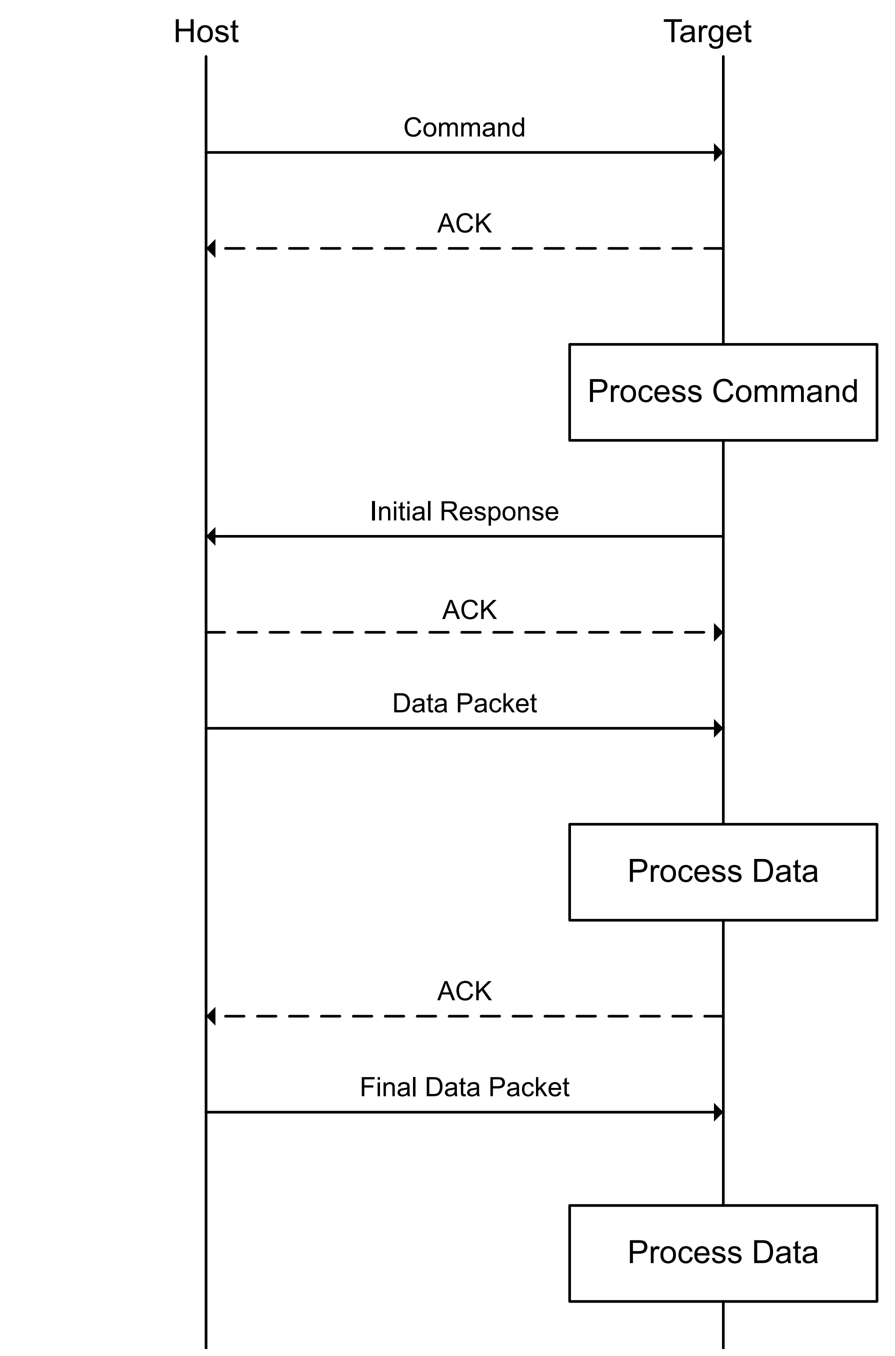


Fig 20. Packet flow command with incoming data phase

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).

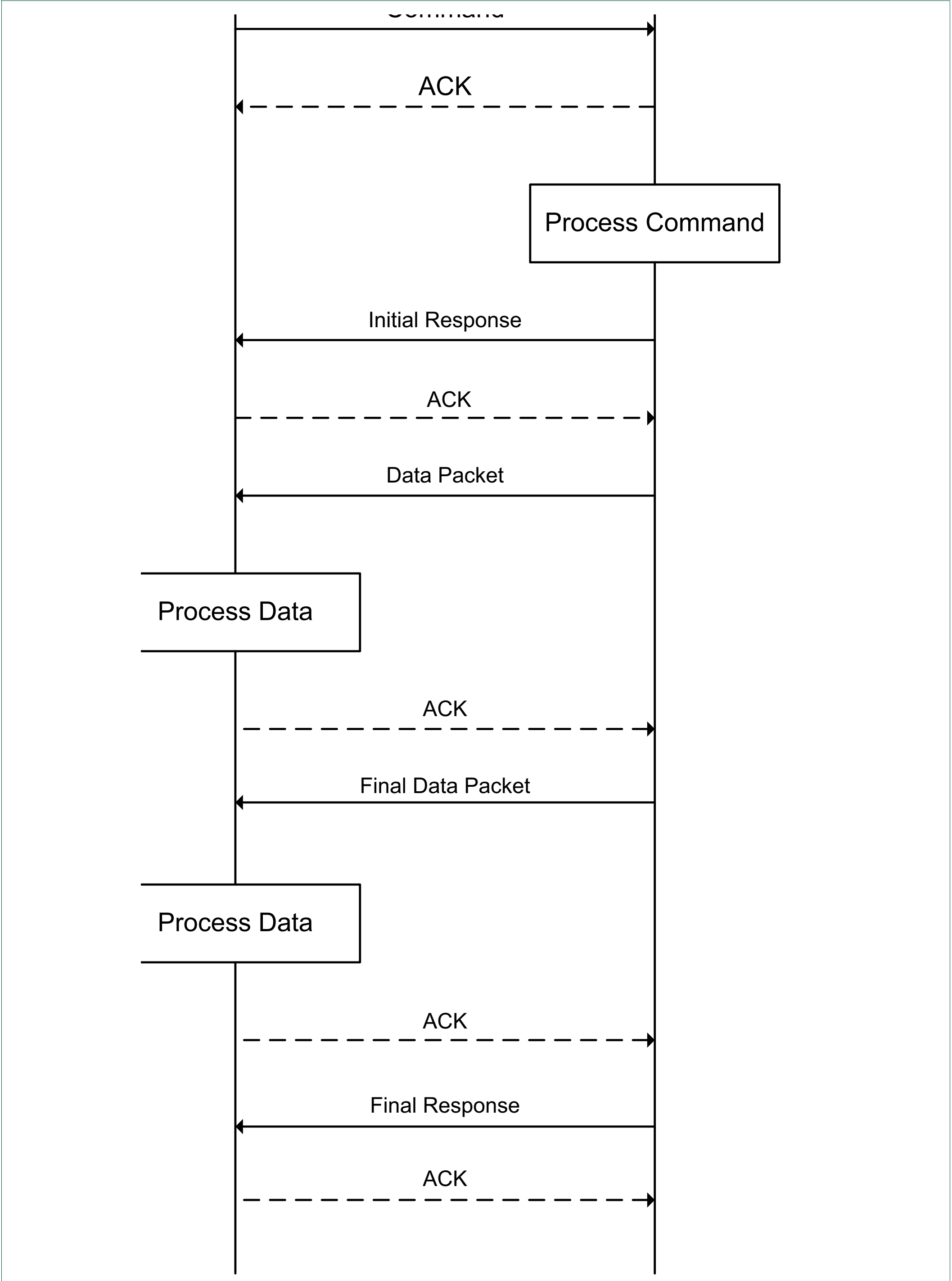


Fig 21. Command with outgoing data phase

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 212. Ping packet format

Byte #	Value	Name
0	0x5A	Start byte
1	0xA6	Ping

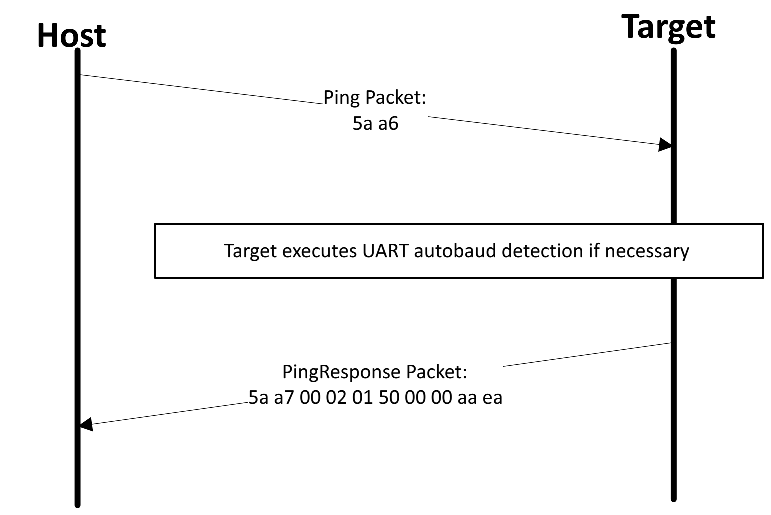

Host
Target
Ping Packet:
5a a6
Target executes UART autobaud detection if necessary
PingResponse Packet:
5a a7 00 02 01 50 00 00 aa ea

Fig 22. 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 213. 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^{2}C, and SPI.

Table 214. 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 215. 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 216. 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.
0xA4	kFramingPacketType_Command	The framing packet contains a command packet payload.

Table 216. Packet type field

Packet type	Name	Description
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 217. 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 218. 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 219. 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 220. 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. [1]
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	

[1]The GetProperty, Reset, KeyProvisioning, SetProperty, and ReceiveSbFile are allowed in limited ISP mode (CMPA_DIGEST written).

Table 221. 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 222. 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 223. 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 224. 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 225. 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 226. 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 8.6.13 “KeyProvision command”.

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 227. 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 is equal to 8).

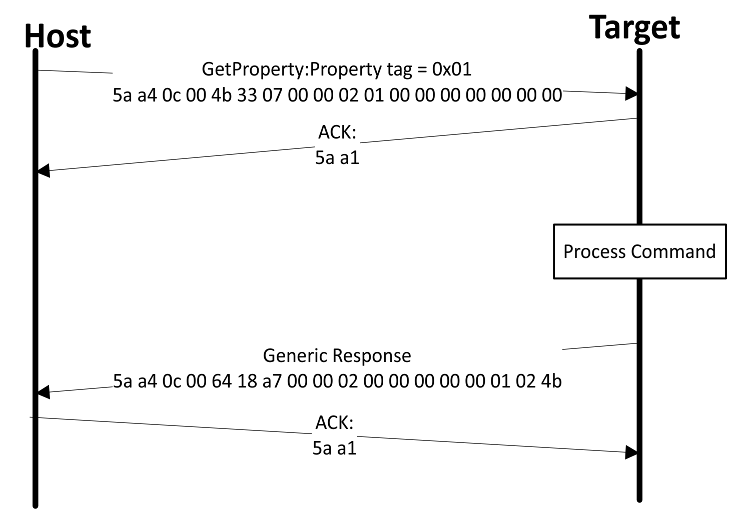

Host
Target
GetProperty:Property tag = 0x01
5a a4 0c 00 4b 33 07 00 00 02 01 00 00 00 00 00 00 00
ACK:
5a a1
Process Command
Generic Response
5a a4 0c 00 64 18 a7 00 00 02 00 00 00 00 00 01 02 4b
ACK:
5a a1

Fig 23. Protocol sequence for GetProperty command

Table 228. GetProperty command packet format (example)

GetProperty	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x0C 0x00
	Crc16	0x4B 0x33

Table 228. GetProperty command packet format (example)

GetProperty	Parameter	Value
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 229 shows an example of a GetPropertyResponse packet.

Table 229. 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 230. Parameters for SetProperty command

Byte #	Command
0 - 3	Property tag
4 - 7	Property value

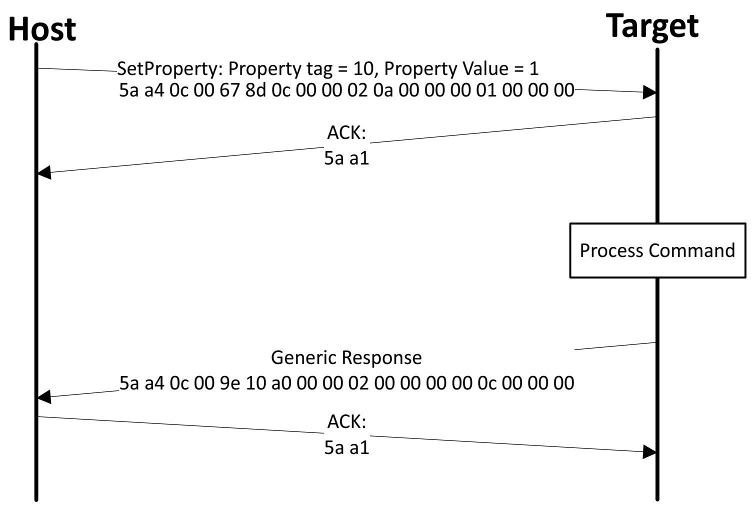

Host
Target
SetProperty: Property tag = 10, Property Value = 1
5a a4 0c 00 67 8d 0c 00 00 02 0a 00 00 00 01 00 00 00
ACK:
5a a1
Process Command
Generic Response
5a a4 0c 00 9e 10 a0 00 00 02 00 00 00 00 0c 00 00 00
ACK:
5a a1

Fig 24. Protocol sequence for SetProperty Command

Table 231. 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 232. 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 233. Parameter for FlashEraseAll command

Byte #	Parameter	
0-3	Memory ID	
	0x000	Internal Flash
	0x110	Serial NOR/EEPROM through SPI

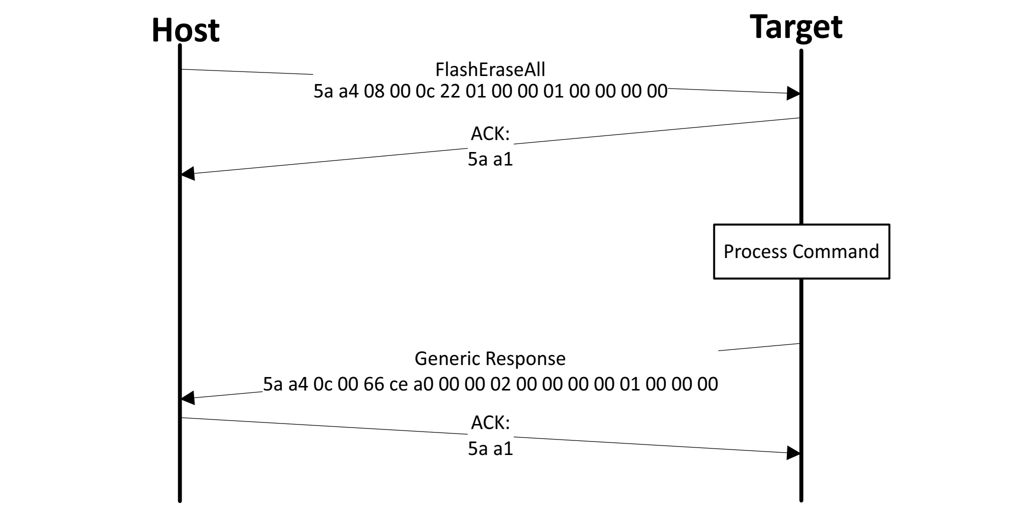

Host
Target
FlashEraseAll
5a a4 08 00 0c 22 01 00 00 01 00 00 00 00
ACK:
5a a1
Process Command
Generic Response
5a a4 0c 00 66 ce a0 00 00 02 00 00 00 00 01 00 00 00
ACK:
5a a1

Fig 25. Protocol sequence for FlashEraseAll command

Table 234. FlashEraseAll command packet format (example)

FlashEraseAll	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x08 0x00
	Crc16	0x0C 0x22
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 235. 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.

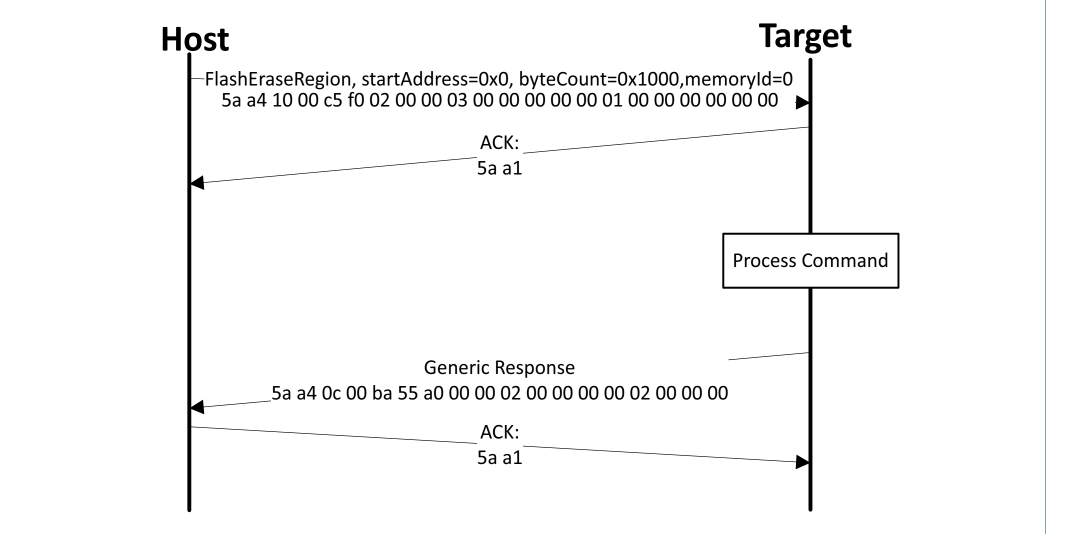

Host
Target
FlashEraseRegion, startAddress=0x0, byteCount=0x1000,memoryId=0
5a a4 10 00 c5 f0 02 00 00 03 00 00 00 00 00 01 00 00 00 00 00 00
ACK:
5a a1
Process Command
Generic Response
5a a4 0c 00 ba 55 a0 00 00 02 00 00 00 00 02 00 00 00
ACK:
5a a1

Fig 26. Protocol sequence for FlashEraseRegion command

Response: The target returns a GenericResponse packet with one of the following error status codes.

Table 236. FlashEraseRegion response status codes

Status code
kStatus_Success (0).
kStatus_MemoryRangeInvalid (10200).
kStatus_FlashAlignmentError (101).
kStatus_IFlashAddressError (102).

Table 236. FlashEraseRegion response status codes

Status code
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 237. 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.

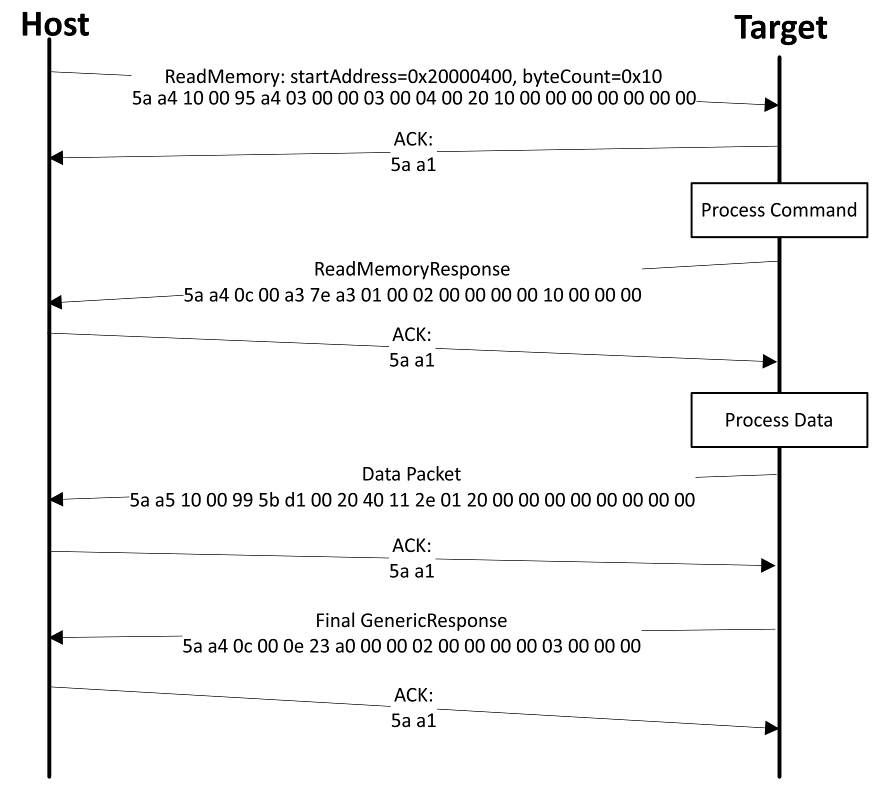

Host
Target
ReadMemory: startAddress=0x20000400, byteCount=0x10
5a a4 10 00 95 a4 03 00 00 03 00 04 00 20 10 00 00 00 00 00 00 00 00
ACK:
5a a1
Process Command
ReadMemoryResponse
5a a4 0c 00 a3 7e a3 01 00 02 00 00 00 00 10 00 00 00
ACK:
5a a1
Process Data
Data Packet
5a a5 10 00 99 5b d1 00 20 40 11 2e 01 20 00 00 00 00 00 00 00 00 00
ACK:
5a a1
Final GenericResponse
5a a4 0c 00 0e 23 a0 00 00 02 00 00 00 00 03 00 00 00
ACK:
5a a1

Fig 27. Command sequence for ReadMemory

Table 238. 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 239. Parameters for WriteMemory command

Byte #	Command
0-3	Start address
4-7	Byte count
8-11	Memory ID

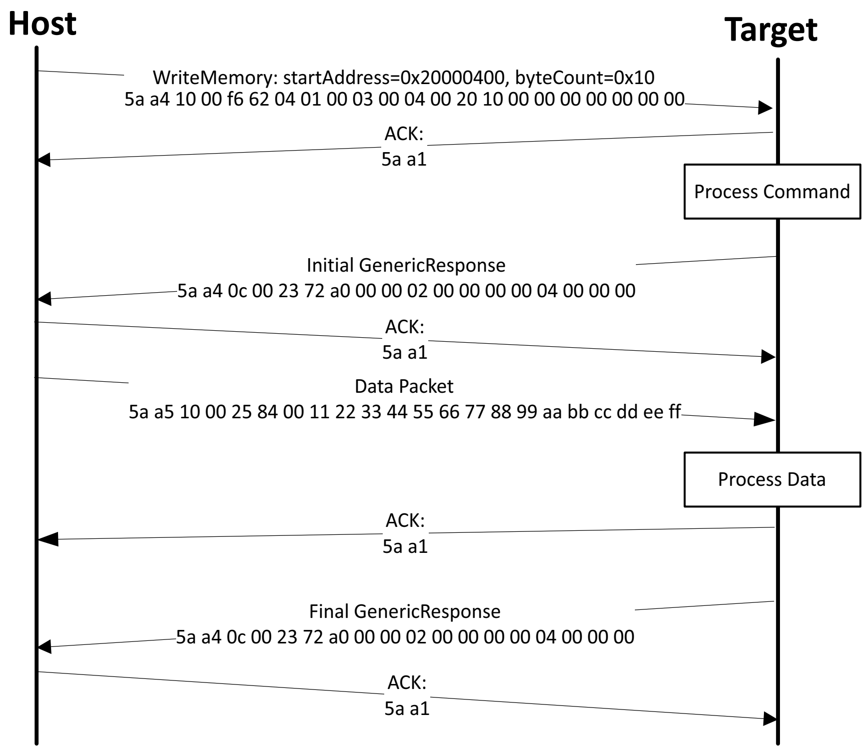

Host
Target
WriteMemory: startAddress=0x20000400, byteCount=0x10
5a a4 10 00 f6 62 04 01 00 03 00 04 00 20 10 00 00 00 00 00 00 00
ACK:
5a a1
Process Command
Initial GenericResponse
5a a4 0c 00 23 72 a0 00 00 02 00 00 00 00 04 00 00 00
ACK:
5a a1
Data Packet
5a a5 10 00 25 84 00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
Process Data
ACK:
5a a1
Final GenericResponse
5a a4 0c 00 23 72 a0 00 00 02 00 00 00 00 04 00 00 00
ACK:
5a a1

Fig 28. Protocol sequence for WriteMemory command

Table 240. 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 241. 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.

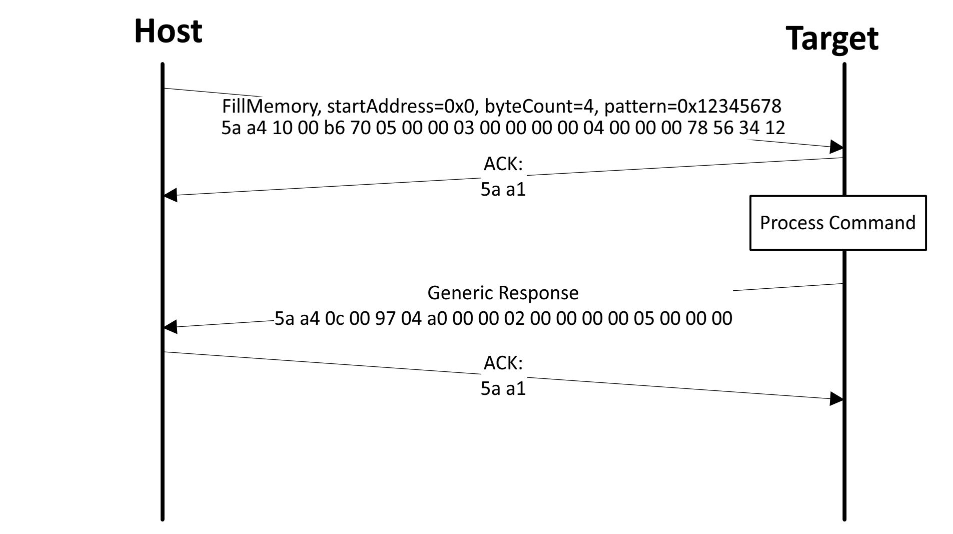

Host
Target
FillMemory, startAddress=0x0, byteCount=4, pattern=0x12345678
5a a4 10 00 b6 70 05 00 00 03 00 00 00 00 04 00 00 00 78 56 34 12
ACK:
5a a1
Process Command
Generic Response
5a a4 0c 00 97 04 a0 00 00 02 00 00 00 00 05 00 00 00
ACK:
5a a1

Fig 29. Protocol sequence for FillMemory command

Table 242. 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 243. 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 Reset command

The Reset command results in the bootloader resetting the chip.

The Reset command requires no parameters.

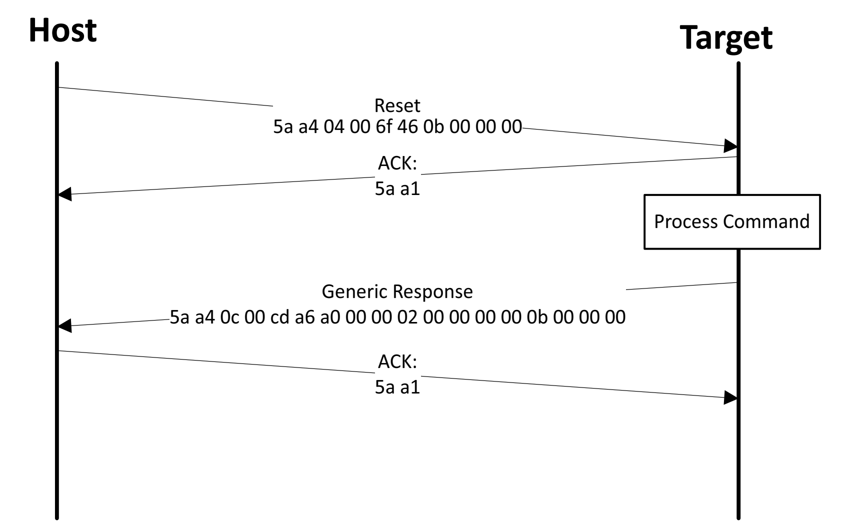

Host
Target
Reset
5a a4 04 00 6f 46 0b 00 00 00
ACK:
5a a1
Process Command
Generic Response
5a a4 0c 00 cd a6 a0 00 00 02 00 00 00 00 0b 00 00 00
ACK:
5a a1

Fig 30. Protocol sequence for Reset command

Table 244. 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 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 245. 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.

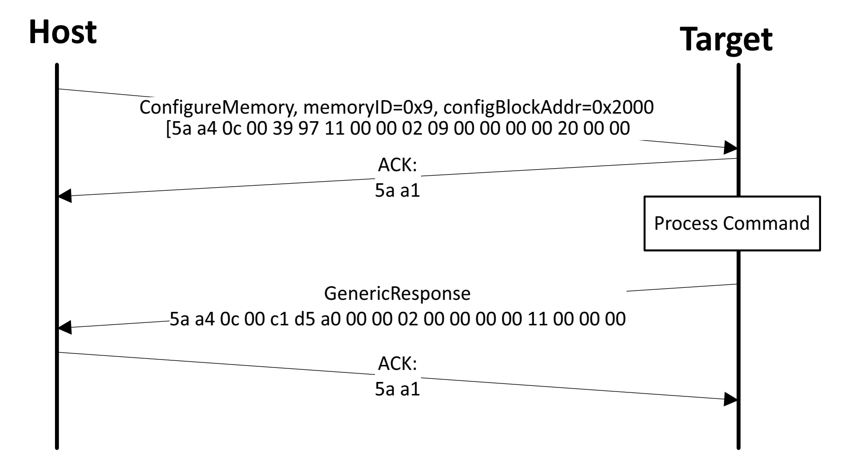

Host
Target
ConfigureMemory, memoryID=0x9, configBlockAddr=0x2000
[5a a4 0c 00 39 97 11 00 00 02 09 00 00 00 00 20 00 00
ACK:
5a a1
Process Command
GenericResponse
5a a4 0c 00 c1 d5 a0 00 00 02 00 00 00 00 11 00 00 00
ACK:
5a a1

Fig 31. Protocol sequence for ConfigureMemory command

8.6.11.1 Supported Memory IDs

The following table shows the supported memory IDs.

Table 246. 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

The boot ROM supports programming 1-bit SPI NOR FLASH devices (which supports a 3-byte address read 0x03 or 4-byte address read 0x13) via the Flash Configuration Option Block. The SPI clock frequency is set to 24Mhz by the ROM.

The boot ROM supports either a manually configured option or an auto-detected option. When using the manually configured option, you must specify all the Flash information (Flash size, sector size, page size) in the option block. When using the auto-detected option, (which is only supported on devices that are JESD216 compliant) the boot ROM is able to detect the Flash information via the read SFDP (0x5A) command, where you can set all the Flash information to 0x0s. Table 247 shows the details required for configuring SPI NOR FLASH using either of these methods. The read Page (0x03) is used for the default read command, if the FLASH size is greater than 16MB (detected by the read SFDP command), the 0x13 command is used for the page read.

If the SFDP command is not supported, ROM uses the read MID (0x9F) to detect whether there is a connected device to the LPC55S0x/LPC550x.

Table 247. 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	0 - 512KB	0 - 4KB	0 - 256 Bytes
			(Select Flash Parameter via Flash Datasheet)	1 - 1MB	1 - 8KB	1 - 512 Bytes
				2 - 2MB	2 - 32KB	2 - 1KB
			2 - Auto	3 - 4MB	3 - 64KB	3 - 128KB
			(Detect Flash Parameter via SFDP table)	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

This example uses the manually configured parameter.

The FLASH page size is 256-byte, sector size is 4KB, Flash size is 512KB. The configuration option block is 0xc000_0000. 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 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 248. 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.

8.6.13 KeyProvision command

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 249. 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 249. 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 250 and Table 251 describes the details of each KeyProvision operation and Key Type.

Table 250. 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.

Table 250. KeyProvision operation details

Value	Operation	Details
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 251. Key Type details

Value	Key Type
0x0	Invalid
0x1	HashCrypt SRK
0x2	Invalid
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	UDS

Command: KeyProvision command packet format is shown in Table 252.

Table 252. 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 250)
	key type / memory ID	0x00000000 (see Table 251)
	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 253 describes the KeyProvisionResponse packet.

Table 253. 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 Get/SetProperty command properties

This section lists the properties of the GetProperty and SetProperty commands.

Table 254. 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

Table 254. Properties used by Get/SetProperty commands, sorted by values *…continued*

Property	Writable	Tag Value	Size	Description
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
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	The life cycle of the device. 0x5aa55aa5 - Device is in development life cycle. 0xc33cc33c - Device is in deployment life cycle.

Table 254. 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 Property definitions

Get/Set property definitions are provided in this section.

8.6.14.1.1 CurrentVersion property

The value of this property is a 4-byte structure containing the current version of the bootloader.

Table 255. CurrentVersion property fields

Bit	[31:24]	[23:16]	[15:8]	[7:0]
Field	Name = 'K' (0x4B)	Major version	Major version	Bugfix version

8.6.14.1.2 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 256. Peripheral bits

Bit	[31:7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Field	Reserved	Reserved	Reserved	Reserved	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.

8.6.14.1.3 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 257. 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

8.6.14.1.4 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 258. 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.

8.6.14.1.5 GetLastError property

The following table lists the response words and corresponding error conditions.

Table 259. 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	kLog_Auth_Dice_Fail	Dice calculation failed.
0x0d70f300	kLog_Tzm_DeviceMode_Fail	
0x0d71f300	kLog_Tzm_FusesMode_Fail	
0x0d72f300	kLog_Tzm_ImageMode_Fail	

Table 259. Response word and error description

Reponse word	Error	Description
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 260. 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.

Table 260. Bootloader status error codes, sorted by value *…continued*

Error Code	Value	Description
kStatus_FLASH_EncryptedRegionsErase NotDoneAtOnce	134	Encrypted Flash subregions are not erased at once.
kStatus_FLASH_ProgramVerificationNotAll owed	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.
kStatus_FLASH_NmpaAccessNotAllowed	139	The NMPA region is not allowed to be read/written/erased.
kStatus_FLASH_CmpaCfgDirectEraseNot Allowed	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_CfpaVersionRollbackDisa llowed	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.

Table 260. Bootloader status error codes, sorted by value *…continued*

Error Code	Value	Description
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.
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.

Note: In UART, I2C and SPI ISP modes, the LPC55xx expects posted responses to be read by the host within 20ms*number of bytes, otherwise the LPC55xx reports the abort command. Considering an ACK from the LPC55xx is only two bytes, the host must read those bytes within 40ms of the LPC55xx posting.

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.

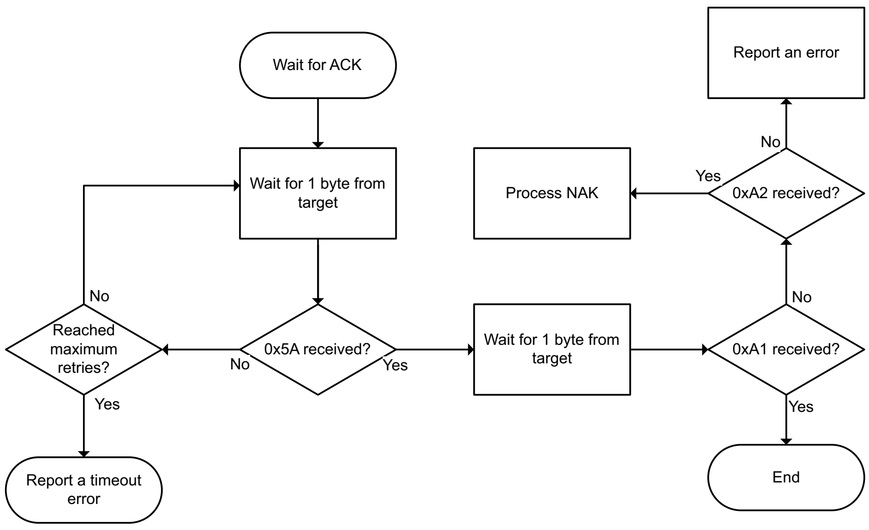


Fig 32. Host reads an ACK from target via UART

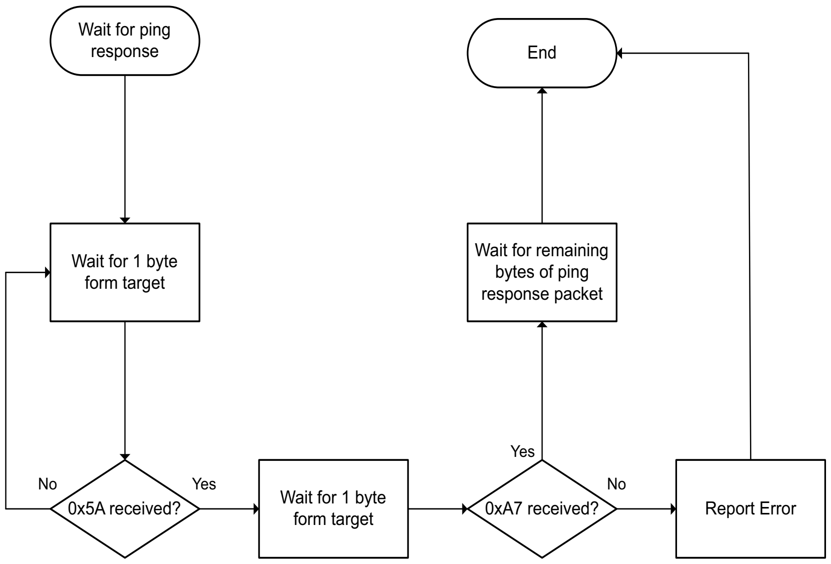

Wait for ping response
Wait for 1 byte form target
0x5A received?
No
Yes
Wait for 1 byte form target
0xA7 received?
Yes
No
Wait for remaining bytes of ping response packet
End
Report Error

Fig 33. Host reads a ping response from target via UART

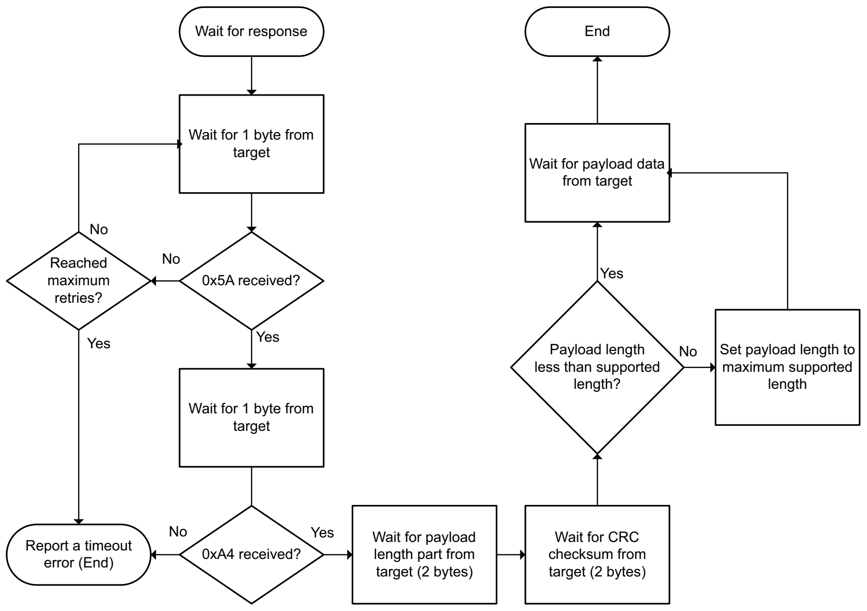

Wait for response
Wait for 1 byte from target
0x5A received?
No
Reached maximum retries?
No
Yes
Yes
Wait for 1 byte from target
0xA4 received?
No
Yes
Report a timeout error (End)
Wait for payload length part from target (2 bytes)
Wait for CRC checksum from target (2 bytes)
Payload length less than supported length?
Yes
No
Set payload length to maximum supported length
Wait for payload data from target
End

Fig 34. 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 I^{2}C In-System Programming

8.9.1 Introduction

The bootloader supports loading data into flash via the I^{2}C peripheral, where the I^{2}C peripheral serves as the I^{2}C slave. A 7-bit slave address is used during the transfer. The bootloader uses 0x10 as the I^{2}C slave address and supports up to 400 kbit/s as the I^{2}C baud rate.

The maximum supported I^{2}C 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 I^{2}C peripheral serves as an I^{2}C 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 I^{2}C slave address and the direction bit is set to write.
- An outgoing packet is read by the host with a selected I^{2}C 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.

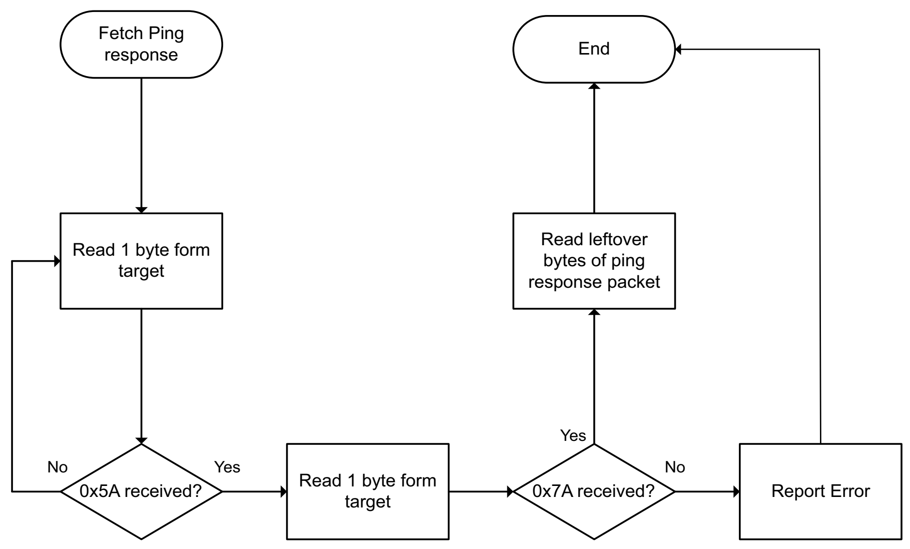


Fig 35. Host reads ping response from target via I^{2}C

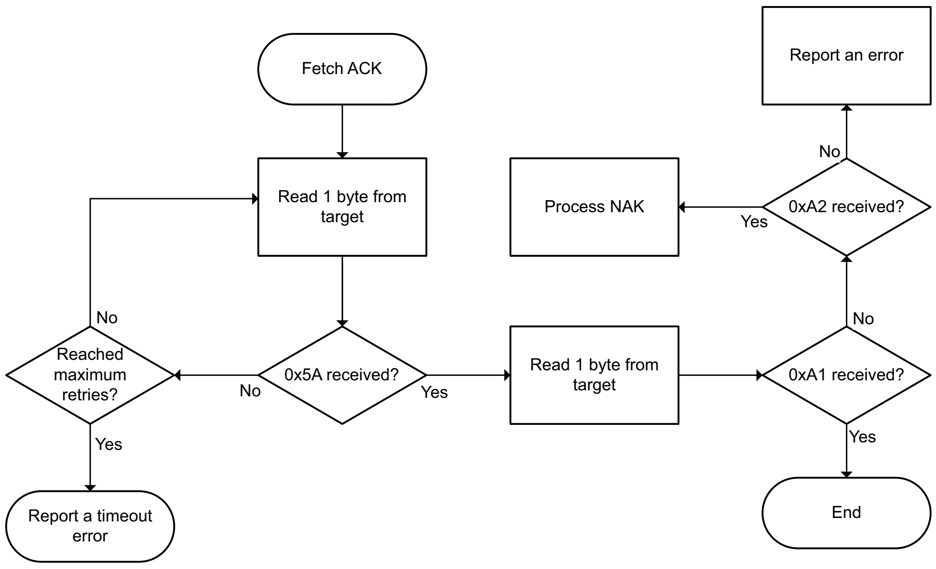


Fig 36. Host reads ACK packet from target via I^2C

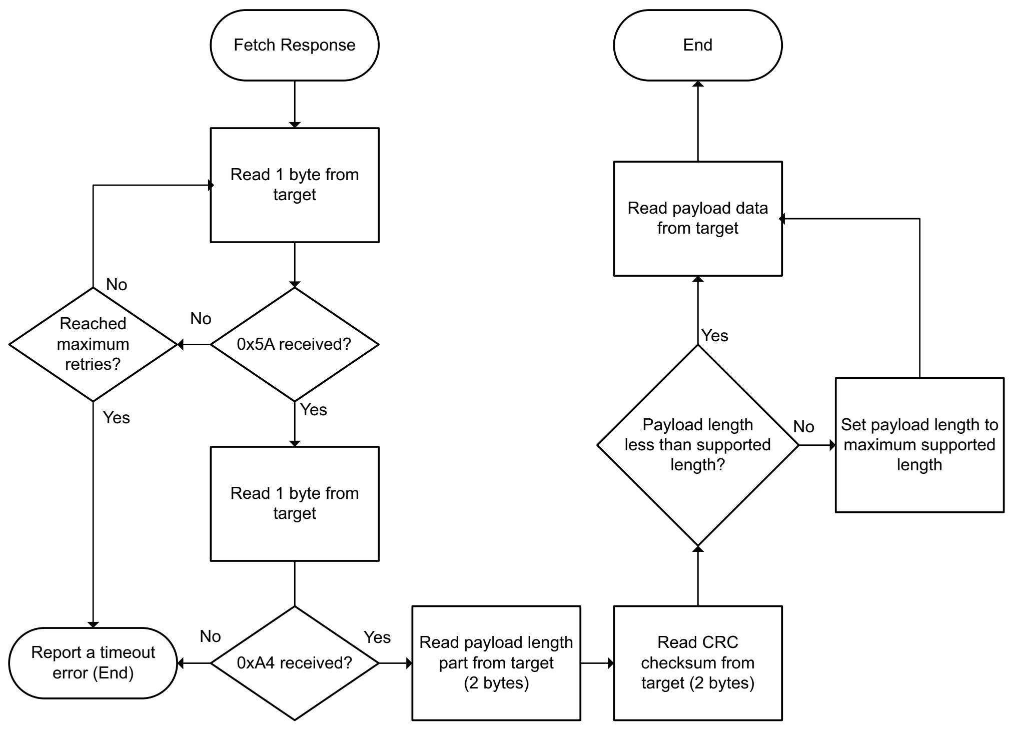

Fetch Response
Read 1 byte from target
No
Reached maximum retries?
No
0x5A received?
Yes
Yes
Read 1 byte from target
No
Report a timeout error (End)
0xA4 received?
Yes
Read payload length part from target (2 bytes)
Read CRC checksum from target (2 bytes)
End
Read payload data from target
Yes
Payload length less than supported length?
No
Set payload length to maximum supported length

Fig 37. Host reads response from target via I^{2}C

Note: The I2C master needs to read back each byte from the slave in 20ms, otherwise the ROM will abort the transfer.

8.9.2 I^{2}C ISP command format

See Section 8.5 “Bootloader packet types” for more details.

8.9.3 I^{2}C ISP response format

See Section 8.5 “Bootloader packet types” for more details.

8.9.4 I^{2}C ISP data format

See Section 8.5 “Bootloader packet types” for more details.

8.9.5 I^{2}C 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^{2}C:
- 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.

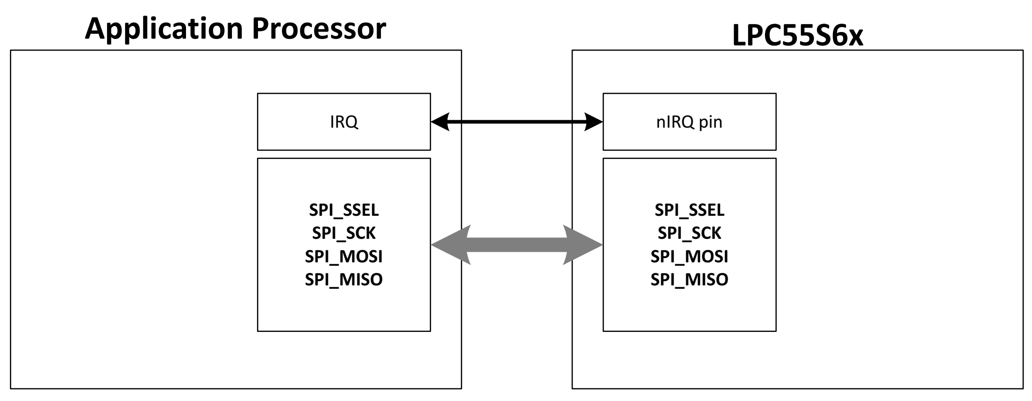


Fig 38. Physical interface for SPI ISP

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.

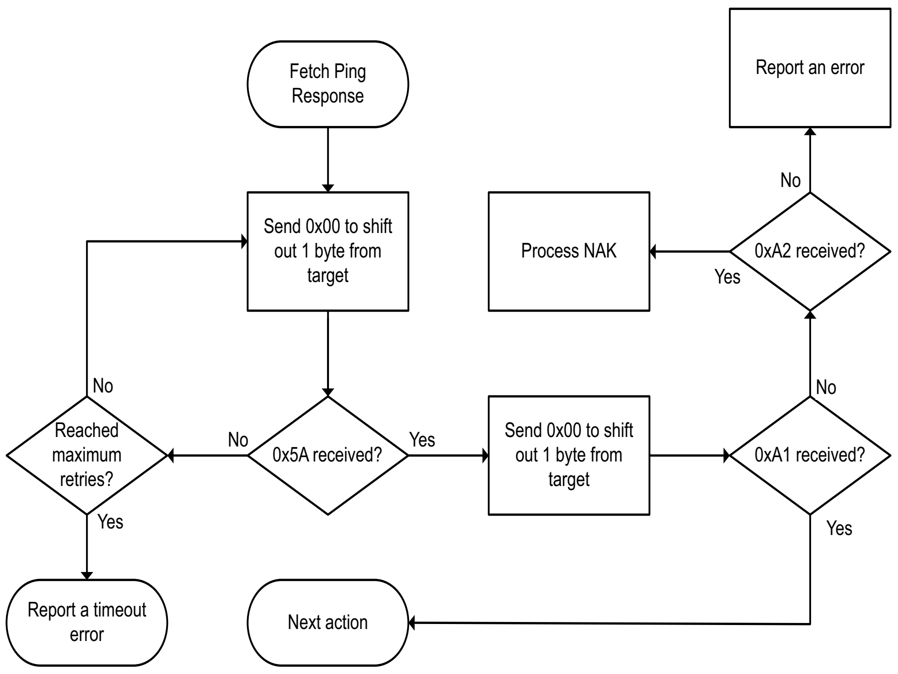


Fig 39. Host reads ping packet from target via SPI

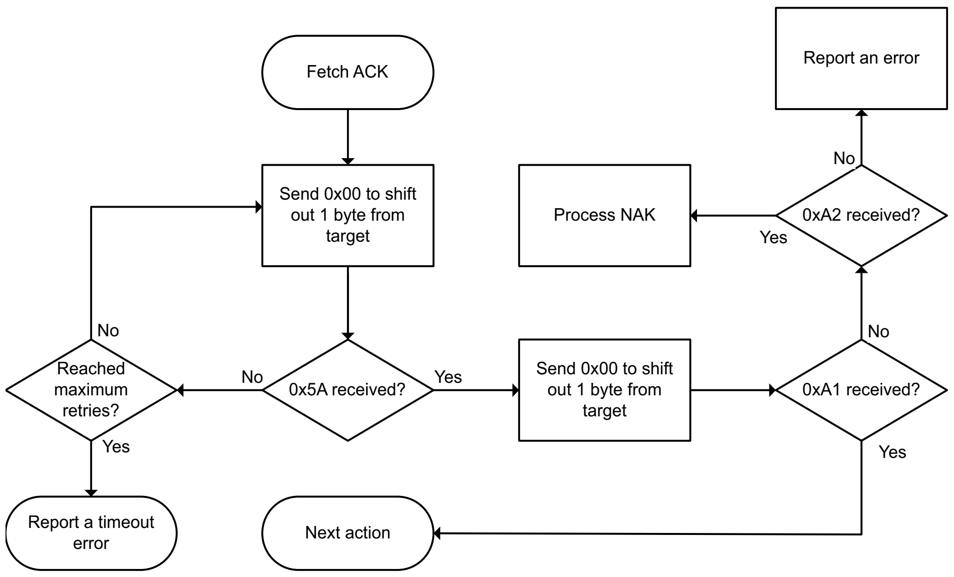


Fig 40. Host reads ACK from target via SPI

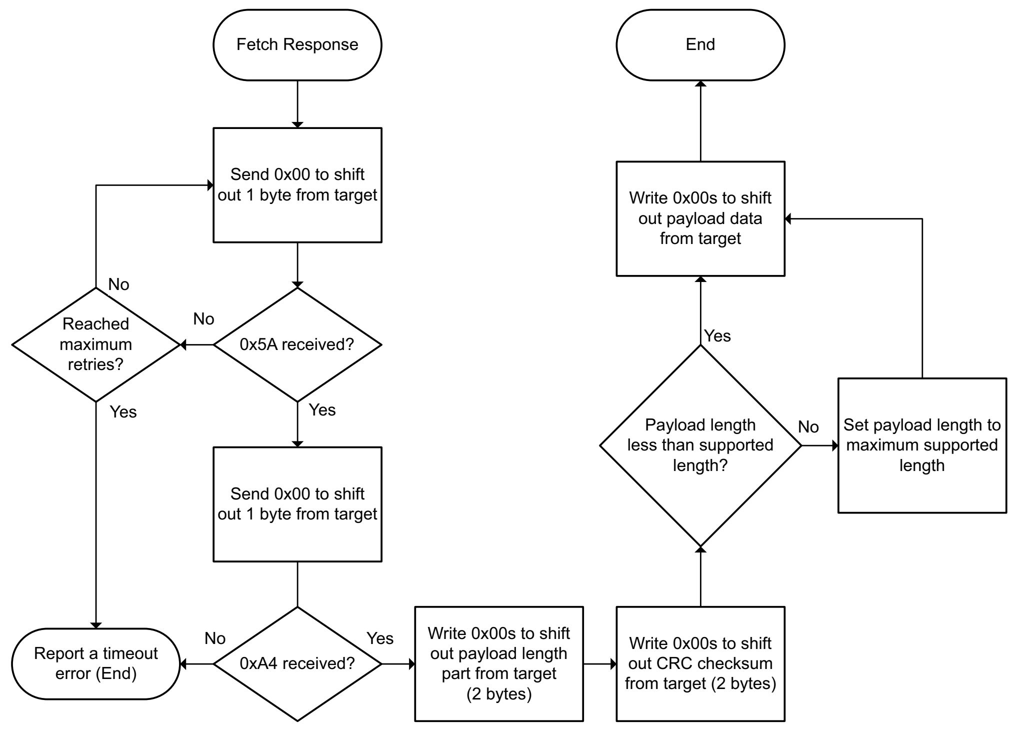

Fetch Response
Send 0x00 to shift out 1 byte from target
0x5A received?
No
Reached maximum retries?
No
Yes
Yes
Send 0x00 to shift out 1 byte from target
0xA4 received?
No
Report a timeout error (End)
Yes
Write 0x00s to shift out payload length part from target (2 bytes)
Write 0x00s to shift out CRC checksum from target (2 bytes)
Payload length less than supported length?
No
Set payload length to maximum supported length
Yes
Write 0x00s to shift out payload data from target
End

Fig 41. 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 In-Application-Programming

See Chapter 9 “LPC55S0x/LPC550x Flash API” for details.

UM11424

Chapter 9: LPC55S0x/LPC550x Flash API

Rev. 1.5 — 21 December 2023 **User manual**

9.1 How to read this chapter

This chapter applies to all LPC55S0x/LPC550x 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 FFR region).
- Boot ROM API support for programming the flash region and the FFR 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 0x1301fe00. See Figure 42 for the ROM API layout.

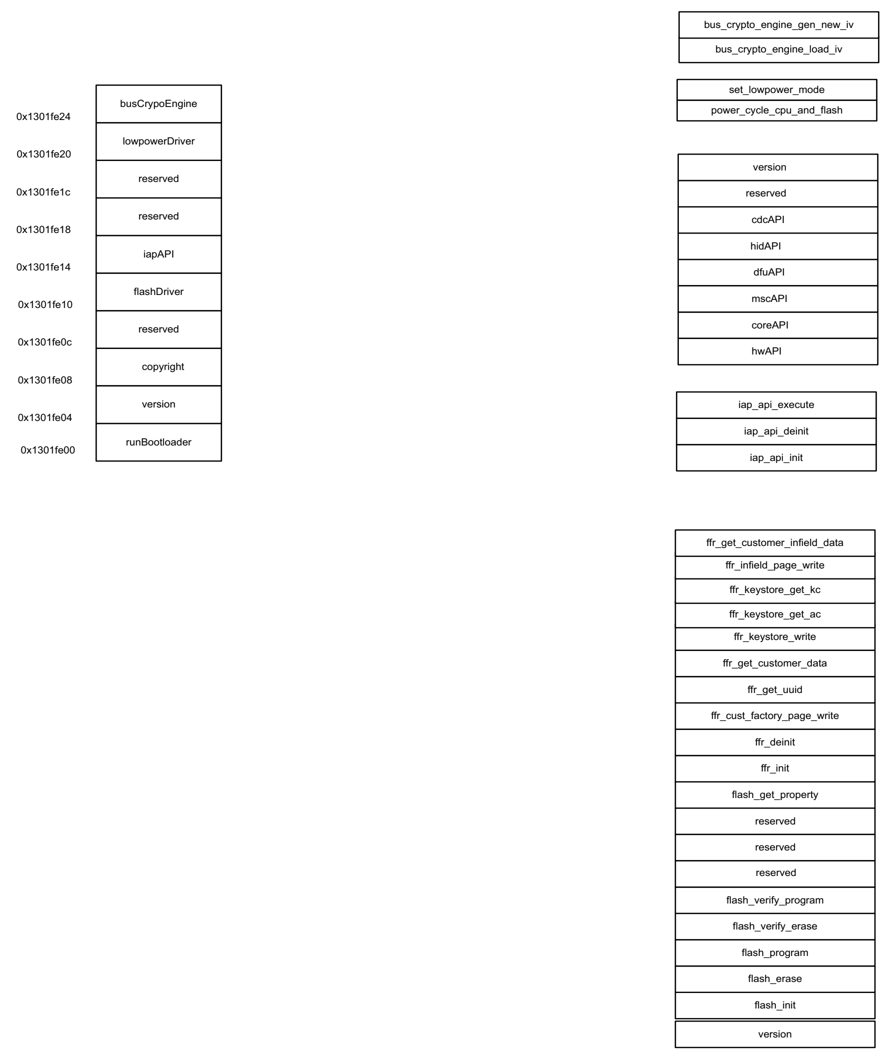


Fig 42. ROM API structure

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];
    // Flash FFR 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 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 261. 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)0x1301fe00)
#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 “FFR 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 262. 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).
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 “FFR 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 263. 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 264. 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 265. 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 266 for supported properties.

Table 266. 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.
kFLASH_PropertyFfrTotalSize	0x41	FFR total size property.
kFLASH_PropertyFfrBlockBaseAddr	0x42	FFR block base address property.
kFLASH_PropertyFfrPageSize	0x43	FFR 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 267. 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.

Table 267. Flash driver status code *…continued*

Status	Code	Description
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 FFR APIs

This section describes each function supported in the FFR 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 FFR controller and the flash_ffr_config context, it must be called before calling other FFR APIs.

Note: flash_init (see: Section 9.3.2.1 “flash_init”) must be called before calling the ffr_init API.

Prototype

```
status = FLASH_API_TREE->flash_init(&flashConfig);
status_t ffr_init (flash_config_t *config);
```

Table 268. 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 the CFPA, CMPA, and NMPA flash areas. Subsequent writes are inhibited unless a power-on reset (POR) or brown-out detect (BOD) reset occurs.

Prototype

```
status_t ffr_deinit (flash_config_t *config);
```

Table 269. 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 270. 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.

Prototype

```
status_t ffr_get_uuid (flash_config_t *config, uint8_t* uuid)
```

Table 271. 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 272. 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 273. 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 274. 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 275. 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 276. 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 277. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
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 );
```

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 278. 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 - Reserved
		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.

UM11424

Chapter 10: LPC55S0x Protected Flash Region

Rev. 1.5 — 21 December 2023 User manual

10.1 How to read this chapter

The Protected Flash Region is included on all LPC55S0x devices

10.2 General description

The LPC55S0x family consists of 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 as described in the following sections.

10.2.1 Customer Manufacturing Programmable Area (CMPA)

In this region, the user can set the following features:

- Boot Configuration: Defines the boot speed where the Core clock can be set to 48MHz FRO or 96MHz FRO. Default ISP mode (Auto ISP, UART ISP, SPI Slave ISP, I2C slave ISP, and Disable ISP). Boot Failure Indication using GPIO port pin (default is PIO0_0).
- 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-1 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, 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.

  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 so the vendor can sub-divide the VU attribute bit-wise, thus a combination of these different interpretations could be supported.

- Secure boot flags (SECURE_BOOT_CFG) allows a 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), Universal Device Secret (UDS key code) for DICE (HMAC-SHA256-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 a subsequent boot, the ROM checks that the scratch page is valid and has a 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 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 run-time code.

10.3 LPC55S0x Customer Development Lifecycle state

Customers will initially use the LPC55S0x device in the “NXP closed” lifecycle state. Depending on application security requirements, customers can ship the LPC55S0x in this “NXP closed” state or in 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 LPC55S0x, not before erasing various secrets that were provisioned by NXP and Tier1/2/OEM. This state allows FA testing to be run 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, and their impact on each subsystem) and transitioning between states (PFR fields that need to be programmed) are provided in the table below:

Table 279. 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 programmable: 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 • UDS key • SBKEK • User KEK Optional: • Customer defined fields • CFPA fields	CMPA_DIGEST (SHA256 hash of CMPA pages) are left blank.	Tier 1 Dev	-

Table 279. 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 default state, then limited ISP comands are allowed.	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 • UDS key • SBKEK • User KEK Optional: Customer defined fields	• CMPA_DIGEST is programmed, but Debug Authentication configurations 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 configurations 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 279. 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. • Limited/disabled ISP commands.	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. • Limited/disabled ISP commands. 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.

Table 279. Lifecycle state descriptions

Life cycle state	Descriptions	PFR Fields to provision	Transitions		
			Fields to Write	Next State	Notes
	Returned (CQI): • ROM checks that the key store is empty and blocks PUF key unwrapping functionality before enabling Test/Debug ports. • 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 to load and run code in RAM using the 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.		

Chapter 11: LPC55S0x/LPC550x Analog control

11.1 How to read this chapter

The analog controller is available on all LPC55S0x/LPC550x 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. The FRO is trimmed to +/- 1% accuracy over the entire voltage and 0 C to 85 C. The FRO is trimmed to +/- 2% accuracy over the entire voltage and -40 C to 105 C.
- 32 kHz internal FRO.
- High-accuracy frequency measurement function for on-chip and off-chip clocks.
- High-speed Crystal oscillator module control and status (from 16 MHz to 32 MHz).
- 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.

11.3 Basic configuration

- Set the ANALOG_CTRL bit in the AHBCLKCTRL2 register Section 4.5.18 “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 will be automatically enabled when: the RTC_OSP_PD bit in the RTC control register = 0 and the SEL bit in the 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.

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 283.

11.4 Pin description

The analog controller has no configurable pins.

11.5 Register description

Table 280. Register overview: ANACTRL (base address = 0x50013000) bit description

Name	Access	Offset	Description	Reset value	Section
ANALOG_CTRL_CFG [1]	RW	0x0	Various Analog blocks configuration (like FRO 192MHz trimmings source...).	0x0000 0000	11.5.1
ANALOG_CTRL_STATUS	RO	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	WO	0x14	FRO 192 MHz status register.	0x0000 0003	11.5.5
ADC_CTRL [1]	RW	0x18	ADC control static configuration.	0x0000 0000	11.5.6
XO32M_CTRL	RW	0x20	High-speed crystal oscillator control register.	0x0021 428A	11.5.7
XO32M_STATUS	RO	0x24	High-speed 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	RO	0x34	BoDs & DCDC interrupts status register.	0x0000 012D	11.5.10
RINGO0_CTRL [1]	RW	0x40	First Ring Oscillator module control register.	0x0000 0040	11.5.11
RINGO1_CTRL [1]	RW	0x44	Second Ring Oscillator module control register.	0x0000 0040	11.5.12
RINGO2_CTRL [1]	RW	0x48	Third Ring Oscillator module control register.	0x0000 0040	11.5.13
LDO_XO32M [1]	RW	0xB0	High Speed Crystal Oscillator (16 MHz - 32 MHz) Voltage Source Supply Control register.	0x0000 03A0	11.5.14
AUX_BIAS [1]	RW	0xB4	Auxiliary bias.	0x0007 03A0	11.5.15
DUMMY_CTRL	RW	0xF8	Dummy Control Bus to analog modules.	0x0000 0000	11.5.16

[1] 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.

11.5.1 Analog control configuration register [1]

The analog control configuration register is used to configure various analog blocks.

Table 281. (ANALOG_CTRL_CFG, offset = 0x0) bit description

Bit	Symbol	Access	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	-	WO		Reserved. Read value is undefined, only zero should be written	undefined

[1] 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.

11.5.2 Analog control status register

The analog control status register gathers some information related to various analog modules (Flash status and PMU Identification number).

Table 282. (ANALOG_CTRL_STATUS, offset = 0x4) bit description

Bit	Symbol	Value	Description	Reset value
11:0	-	-	Reserved. Read value is undefined.	0x0
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

This 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 283. (FREQ_ME_CTRL, offset = 0xC) bit description

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 ref counter counts to during measurement. Note that the minimum count is 2 ie $2^2 = 4$.	0x0
31	PROG		Set this bit to one 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) - See: Section 11.6.5.
- Frequency target clock select register (FREQMEAS_TARGET) - See: Section 11.6.6.

11.5.4 FRO192M control register

This register is used to configure the on-chip high-speed Free Running Oscillator (FRO192 MHz).

Table 284. FRO 192M control register (FRO192M_CTRL, offset = 0x10) bit description

Bit	Symbol	Access	Value	Description	Reset value
5:0	BIAS_TRIM	RW		Bias trimming bits (course frequency trimming).	0x1
6	-	WO		Reserved. Read value is undefined, only zero should be written.	-
13:7	TEMP_TRIM	RW		Temperature coefficient trimming bits.	-
14	ENA_12MHZCLK	RW		12 MHz clock control.	0x1
			0	12 MHz clock is disabled.	
			1	12 MHz clock is enabled.	
15	-	RW		Reserved. Only 1 should be written. Writing zero prevents the Flash from working.	0x1
23:16	DAC_TRIM	RW	-	Frequency trim.	0x80
25:24	-	WO	-	Reserved. Read value is undefined, only zero should be written.	0x0
29:26	-	RW	-	Reserved. Read value is undefined, only zero should be written.	undefined.
30	ENA_96MHZCLK	RW		96 MHz clock control.	0x0
			0	96 MHz clock is disabled.	
			1	96 MHz clock is enabled.	
31	WRTRIM	WO		This must be written to 1 to modify the BIAS_TRIM and TEMP_TRIM fields.[1]	0x0

[1] Disclaimer: The values 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.

11.5.5 FRO192M status register

High Speed Free Running Oscillator (FRO) Status Register.

Table 285. FRO 192M status register (FRO192M_STATUS, offset = 0x14) bit description

Bit	Symbol	Value	Description	Reset value
0	CLK_VALID		Output clock valid signal.	0x1
		0	No output clock present (None of 12 MHz, 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 ADC control register [1]

ADC control static configuration register.

Table 286. ADC control static configuration **register (ADC_CTRL, offset = 0x18) bit description**

Bit	Symbol		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	-	WO		Reserved. Read value is undefined, only zero should be written	undefined

[1] 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.

11.5.7 High-speed crystal oscillator control register

High-speed crystal oscillator control register.

Table 287. High-speed crystal oscillator control register (XO32M_CTRL, offset = 0x20) bit description

Bit	Symbol	Access	Value	Description	Reset value
0		WO		Reserved. Read value is undefined, only zero should be written.	-
3:1	GM [1]	RW		Gm value for Xo.	0x5
4	SLAVE [1]	RW		Xo in slave mode.	0x0
7:5	AMP [1]	RW		Amplitude selection , Min amp : 001, Max amp : 110.	0x4
14:8	OSC_CAP_IN [1]	RW		Tune capa banks of High speed Crystal Oscillator input pin	0x42
21:15	OSC_CAP_OUT [1]	RW		Tune capa banks of High speed Crystal Oscillator output pin	0x42
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	-	WO		Reserved Read value is undefined, only zero should be written.	0x0
24	ENABLE_SYSTEM_CLK_OUT	RW		Enable High speed Crystal oscillator output to CPU system.	0x0
			0	High speed Crystal oscillator output to CPU system is disabled.	
			1	High speed Crystal oscillator output to CPU system is enabled.	
31:25	-	WO		Reserved. Read value is undefined, only zero should be written.	-

[1] The values written to these bits should not be changed. These bits 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.

11.5.8 High-speed crystal oscillator status register

High Speed Crystal Oscillator (16 MHz - 32 MHz) - also referred as "XO 32 MHz" - output Control Register.

Table 288. 32 MHz Crystal oscillator status register (XO32M_STATUS, offset = 0x24) bit description

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]	0xA
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.	
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] The values written to these bits should not be changed. These bits 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.

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 289. Brown Out Detectors (BoDs) and DCDC interrupts generation control register (BOD_DCDC_INT_CTRL, offset = 0x30) bit description

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 289. Brown Out Detectors (BoDs) and DCDC interrupts generation control register (BOD_DCDC_INT_CTRL, offset = 0x30) *…continued*bit description

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 allows to know the output status and the interrupt status from BoD VBAT, BoD CORE and DCDC

Table 290. BoDs and DCDC interrupts status register (BOD_DCDC_INT_STATUS, offset = 0x34) bit description

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.	0x1
			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.	0x1
			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 290. BoDs and DCDC interrupts status register (BOD_DCDC_INT_STATUS, offset = 0x34) bit description

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 First Ring Oscillator module control register [1]

This register is used to configure the First Ring Oscillator module.

Table 291. First Ring Oscillator module control register (RINGO0_CTRL), offset = 0x40) bit description

Bit	Symbol	Access	Value	Description	Reset value
0	SL	RO		Select short or long rigno (for all ringos types)	0x0
			0	Select short ringo (few elements)	
			1	Select long ringo (many elements)	
1	FS	RW		Ringo frequency output divider	0x0
			0	High frequency output (frequency lower than 100 MHz).	
			1	Low frequency output (frequency lower than 10 MHz).	
3-2	SWN_SWP	RW		PN-Ringos (P-Transistor and N-Transistor processing) control.	0x0
			00	Normal mode.	
			01	P-Monitor mode. Measure with weak P transistor.	
			10	P-Monitor mode. Measure with weak N transistor.	
			11	Don't use	
4	PD	RW		Ringo module power control.	0x0
			0	The Ringo module is enabled.	
			1	The Ringo module is disabled	
5	E_ND0	RW		First NAND2-based ringo control.	0x0
			0	First NAND2-based ringo is disabled.	
			1	First NAND2-based ringo is enabled.	
6	E_NDI	RW		Second NAND2-based ringo control.	0x1
			0	Second NAND2-based ringo is disabled.	
			1	Second NAND2-based ringo is enabled.	
7	E_NR0	RW		First NOR2-based ringo control.	0x0
			0	First NOR2-based ringo is disabled.	
			1	First NOR2-based ringo is enabled.	
8	E_NR1	RW		Second NOR2-based ringo control.	0x0
			0	Second NORD2-based ringo is disabled.	
			1	Second NORD2-based ringo is enabled.	
9	E_IV0	RW		First Inverter-based ringo control.	0x0
			0	First INV-based ringo is disabled.	
			1	First INV-based ringo is enabled.	

Table 291. First Ring Oscillator module control register (RINGO0_CTRL), offset = 0x40) bit description *…continued*

Bit	Symbol	Access	Value	Description	Reset value
10	E_IV1	RW		Second Inverter-based ringo control.	0x0
			0	Second INV-based ringo is disabled.	
			1	Second INV-based ringo is enabled.	
11	E_PN0	RW		First PN (P-Transistor and N-Transistor processing) monitor control.	0x0
			0	First PN-based ringo is disabled.	
			1	First PN-based ringo is enabled	
12	E_PN1	RW		Second PN (P-Transistor and N-Transistor processing) monitor control.	0x0
			0	Second PN-based ringo is disabled.	
			1	Second PN-based ringo is enabled	
15-13	-	WO		Reserved. Read value is undefined, only zero should be written.	-
19-16	DIVISOR	RW		Ringo out Clock divider value. Frequency Output = Frequency input / (DIViSOR+1). (minimum = Frequency input / 16)	0x0
30-20	-	WO		Reserved. Read value is undefined, only zero should be written.	0x0
31	DIV_UPDATE_ REQ	RO		Ringo clock out Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	-

[1] 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.

11.5.12 Second Ring Oscillator module control register [1]

This register is used to configure the Second Ring Oscillator module.

Table 292. First Ring Oscillator module control register (RINGO1_CTRL), offset = 0x44) bit description

Bit	Symbol	Access	Value	Description	Reset value
0	S	RW		Select short or long rigno (for all ringos types)	0x0
			0	Select short ringo (few elements)	
			1	Select long ringo (many elements)	
1	FS	RW		Ringo frequency output divider	0x0
			0	High frequency output (frequency lower than 100 MHz).	
			1	Low frequency output (frequency lower than 10 MHz).	
2	PD	RW		Ringo module Power control.	0x0
			0	The Ringo module is enabled	
			1	The Ringo module is disabled	
3	E_R24	RW			0x0
			0	The Ringo is disabled	
			1	The Ringo is enabled	

Table 292. First Ring Oscillator module control register (RINGO1_CTRL), offset = 0x44) bit description *…continued*

Bit	Symbol	Access	Value	Description	Reset value
4	E_R35	RW			0x0
			0	The Ringo is disabled	
			1	The Ringo is enabled	
5	E_M2	RW		Metal 2 (M2) monitor control.	0x0
			0	The Ringo is disabled	
			1	The Ringo is enabled	
6	E_M3	RW		Metal 3(M3) monitor control.	0x1
			0	The Ringo is disabled	
			1	The Ringo is enabled	
7	E_M4	RW		Metal 4(M4) monitor control.	0x0
			0	The Ringo is disabled	
			1	The Ringo is enabled	
8	E_M5	RW		Metal 5(M5) monitor control.	0x0
			0	The Ringo is disabled	
			1	The Ringo is enabled	
15-9	-	WO		Reserved. Read value is undefined, only zero should be written.	-
19-16	DIVISOR	RW		Ringo out Clock divider value. Frequency Output = Frequency input / (DIViSOR+1). (minimum = Frequency input / 16).	0x0
30-20	-	WO		Reserved. Read value is undefined, only zero should be written.	0x0
31	DIV_UPDATE_ REQ	RO		Ringo clock out Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0x0

[1] 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.

11.5.13 Third Ring Oscillator module control register [1]

This register is used to configure the Third Ring Oscillator module.

Table 293. First Ring Oscillator module control register (RINGO2_CTRL), offset = 0x48) bit description

Bit	Symbol	Access	Value	Description	Reset value
0	S	RW		Select short or long rigno (for all ringos types)	0x0
			0	Select short ringo (few elements)	
			1	Select long ringo (many elements)	
1	FS	RW		Ringo frequency output divider	0x0
			0	High frequency output (frequency lower than 100 MHz).	
			1	Low frequency output (frequency lower than 10 MHz).	

Table 293. First Ring Oscillator module control register (RINGO2_CTRL), offset = 0x48) bit description *…continued*

Bit	Symbol	Access	Value	Description	Reset value
2	PD	RW		Ringo module Power control.	0x0
			0	The Ringo module is enabled	
			1	The Ringo module is disabled	
3	E_R24	RW			0x0
			0	The Ringo is disabled	
			1	The Ringo is enabled	
4	E_R35	RW			0x0
			0	The Ringo is disabled	
			1	The Ringo is enabled	
5	E_M2	RW		Metal 2 (M2) monitor control.	0x0
			0	The Ringo is disabled	
			1	The Ringo is enabled	
6	E_M3	RW		Metal 3(M3) monitor control.	0x1
			0	The Ringo is disabled	
			1	The Ringo is enabled	
7	E_M4	RW		Metal 4(M4) monitor control.	0x0
			0	The Ringo is disabled	
			1	The Ringo is enabled	
8	E_M5	RW		Metal 5(M5) monitor control.	0x0
			0	The Ringo is disabled	
			1	The Ringo is enabled	
15-9	-	WO		Reserved. Read value is undefined, only zero should be written.	-
19-16	DIVISOR	RW		Ringo out Clock divider value. Frequency Output = Frequency input / (DIViSOR+1). (minimum = Frequency input / 16).	0x0
30-20	-	WO		Reserved. Read value is undefined, only zero should be written.	0x0
31	DIV_UPDATE_ REQ	RO		Ringo clock out Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0x0

[1] 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.

11.5.14 High Speed Crystal Oscillator (16 MHz - 32 MHz) Voltage Source Supply Control register [1]

This register controls the LDO bypass for a specified output level.

Table 294. High Speed Crystal Oscillator Voltage Source Supply Control register (LDO_XO32M, offset = 0xB0) bit description

Bit	Symbol	Access	Value	Description	Reset value
0	-	WO	-	Reserved. Read value is undefined, only zero should be written.	-
1	BYPASS	RW		Activates LDO bypass.	0x0
			0	Disables bypass mode (for normal operations).	
			1	Activates LDO bypass.	
2	HIGHZ	RW		Activates LDO bypass.	0x0
			0	Output in High normal state.	
			1	Output in High Impedance state.	
5:3	VOUT	RW		Sets the LDO output level.	0x4
			000	0.750 V.	
			001	0.775 V.	
			010	0.800 V.	
			011	0.825 V.	
			100	0.850 V.	
			101	0.875 V.	
			110	0.900 V.	
			111	0.925 V.	
7:6	IBIAS	RW		Stability configuration.	0x2
9:8	STABMODE	RW		Adjust the biasing current.	0x3
31:10	-	WO		Reserved. Read value is undefined.	-

[1] 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.

11.5.15 Auxiliary bias register [1]

This register controls the output of reference voltage buffer.

Table 295. Auxiliary bias register (AUX_BIAS, offset = 0xB4) bit description

Bit	Symbol	Access	Value	Description	Reset value
0	-	RO	-	Reserved. Read value is undefined, only zero should be written.	-
1	VREF1VENABLE	RW		Controls 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	-	RO		Reserved. Read value is undefined.	-

[1] 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.

11.5.16 Dummy control register

This register provides a control bus to analog module option for high speed crystal oscillator output.

Table 296. Dummy Control bus to analog module register (DUMMY_CTRL, offset = 0xF8) bit description

Bit	Symbol	Access	Value	Description	Reset value
9:0	-	-	-	Reserved. Read value is undefined, only zero should be written.	-
11:10	XO32M_ADC_CLK_MODE	RW		Controls high speed crystal oscillator mod of the ADC clock.	0x1
			0	High speed crystal oscillator output to ADC is disabled.	
			1	High speed crystal oscillator output to ADC is enabled.	
31:12	-	RO		Reserved. Read value is 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 43 shows a block diagram of the frequency measure function.

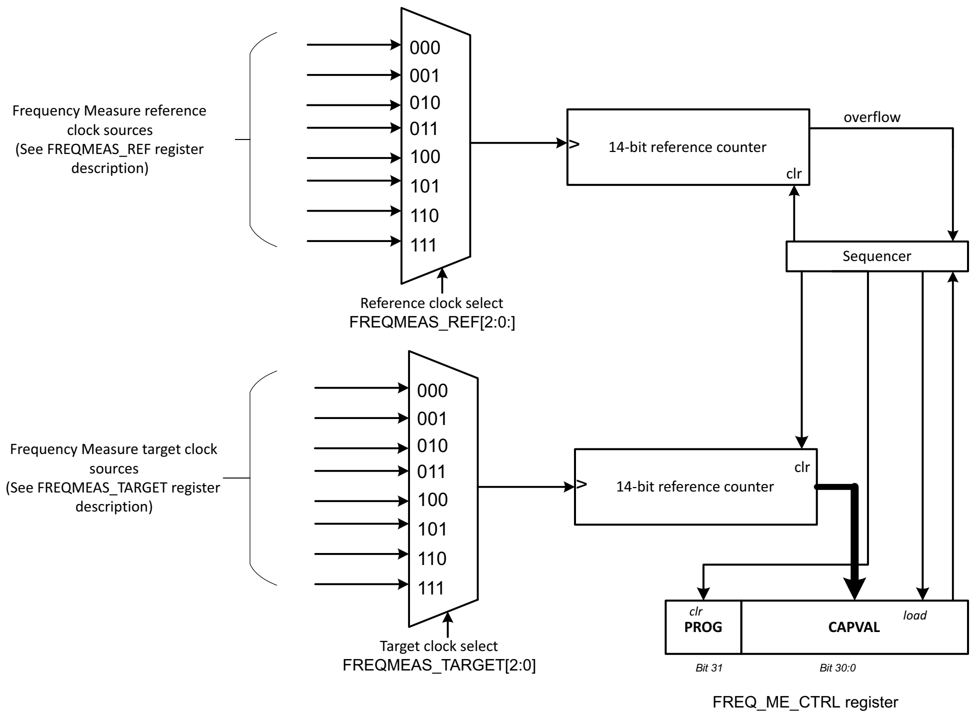

Frequency Measure reference clock sources (See FREQMEAS_REF register description)
000
001
010
011
100
101
110
111
Reference clock select FREQMEAS_REF[2:0:]
14-bit reference counter
overflow
clr
Sequencer
Frequency Measure target clock sources (See FREQMEAS_TARGET register description)
Target clock select FREQMEAS_TARGET[2:0]
14-bit reference counter
clr
clr
PROG
CAPVAL
load
Bit 31
Bit 30:0
FREQ_ME_CTRL register

Fig 43. Frequency measure block diagram

Both target and reference clocks are selectable by programming the target clock select FREQMEAS_TARGET register and reference clock select FREQMEAS_REF register. See Table 388.

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 283.

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 this 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 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 the measuring. 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 to 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 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 some loss of accuracy if the reference frequency exceeds the target frequency by a very large margin (25x or more). Accuracy is not a simple function of the magnitude of the frequency difference, however. 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%.

UM11424

Chapter 12: LPC55S0x/LPC550x Cap Bank API

Rev. 1.5 — 21 December 2023 **User manual**

12.1 How to read this chapter

All LPC55S0x/LPC550x 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, where each can be used as an integrated load capacitor for the crystal oscillators. The capacitor banks on each crystal pin can tune the frequency for crystals with a Capacitive Load (CL) between 6 to 10pF (IEC equivalent).

12.2 Features

- Conserves space on the PCB.
- Reduces the overall Bill of Materials (BOM).
- Accommodates a Capacitive Load (CL) between 6 - 10 pF (IEC equivalent).
- Allows simple APIs to configure the Capacitor Banks based on the crystal Capacitive Load (CL) and measured PCB parasitic capacitances on XIN and XOUT pins.

12.3 Crystal Oscillator Capacitor Banks API description

Table 297. Low power API calls

Function prototype	API description	Section
`void XTAL_16mhz_capabank_trim (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.	
`void XTAL_32khz_capabank_trim (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.	

12.3.1 XTAL_16mhz_capabank_trim

This API function configures the 16 MHz Crystal Oscillator Capacitor Bank.

Table 298. 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 299. 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_32kfXtalIecLoadF_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);

/*

* - (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;
```

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;
```

UM11424

Chapter 13: LPC55S0x/LPC550x Power Management

Rev. 1.5 — 21 December 2023 **User manual**

13.1 How to read this chapter

This chapter provides an overview of power related information about LPC55S0x/LPC550x devices. These devices include a variety of power switches and clock switches to allow fine tuning power usage to match requirements at different performance levels and reduced power modes.

To turn analog components ON or OFF in active mode, use the PMC_PDRUNCFG0 register, see Table 319. 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

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_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, multilayer matrix, and almost all peripherals like Flexcomm, SDMA, 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 300 shows the detailed list of all modules per power domain.

Table 300. 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	-	32-kHz Crystal Oscillator (XTAL32K)		
	PLL1		32-kHz Free Running Oscillator (FRO32K)		
	192-MHz Free Running Oscillator (FRO192M)		1-MHz Free Running Oscillator (FRO1M)		
	High Speed Crystal Oscillator		Analog Comparator		
	ADC		Brown Out Detector VBAT (BoD VBAT)		
	USB High Speed Physical Interface				
	Temperature Sensor				
	Brown Out Detector Core (BoD Core)				
Digital components	CPU0 (Cortex-M33 full feature)	Reset controller	Power Management Controller (PMC)		
	Debug Ports (SWD interface)	I/O Configuration (IOCON)	OS Event Timer		
	Debug Mailbox	I/O functional multiplexers			
	-	Group GPIO Input Interrupt (GINT0/1)			
	CASPER	Flexcomm Interface3			
	CAN				
	PRINCE				
	System DMA				
	Secure System DMA				
	General Purpose I/O Controller				

Table 300. Power domain supply *…continued*

	Power domain core	Power domain system	Power domain AO	Power domain _mem_0	Power domain _mem_1
Digital components	Secure General Purpose I/O Controller				
	High Speed SPI				
	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 (4 KB)
	FLASH (256 KB)			RAM_X3 (4 KB)	RAM_X1 (4 KB)
	PUF RAM (2 KB)				RAM_00 (32 KB)
					RAM_10 (16 KB)
					RAM_20 (16 KB)

Table 300. Power domain supply *…continued*

	Power domain core	Power domain system	Power domain AO	Power domain _mem_0	Power domain _mem_1
Memories					RAM_3 (16 KB)

13.2.3 Power modes

Power modes are controlled exclusively via a power API Chapter 14 "LPC55S0x/LPC550x 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 the CPU 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 state 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 SPI, I^{2}C, 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, where the clock to the CPU and peripherals 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, and all Flexcomm products except for Flexcomm3). Through the power profiles API, selected peripherals such as FLEXCOMM3 Interface, SPI, I^{2}C, 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 (CPU 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 301. 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 302. 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	High Speed Crystal Oscillator	Software configured	Software configured	OFF	OFF
PLL0	1st PLL550M	Software configured	Software configured	OFF	OFF
PLL1	2nd PLL550M	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

Table 302. Peripheral reduced power modes *…continued*

Peripherals		Reduced Power Modes			
Name	**Description**	**Sleep**	**Deep-sleep**	**Power-down**	**Deep-power down**
AUXBIAS	ADC Analog references	Software configured	Software configured	OFF	OFF
LDO_XO_32M	High Speed 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 340 "Parameter wakeup_interrupts".

13.3 Functional description

13.3.1 Power management

The LPC55xx supports a variety of power control features. In active mode, when the chip is running, power and clocks to selected peripherals can be optimized for power consumption. In addition, there are four special modes of processor power reduction with different peripherals running: sleep-mode, deep-sleep mode, power-down and deep-power down mode, activated by the power-mode configure API, see Chapter 14 "LPC55S0x/LPC550x 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 4.5.16 "AHB clock control 0", Section 4.5.17 "AHB clock control 1", and Section 4.5.18 "AHB clock control 2". Generally speaking, in order 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 when they will be needed.
- The power to various analog blocks (PLL, oscillators, and the BOD circuit) can be controlled individually through the PDRUNCFG0 register, see Table 319. 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 2 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 "LPC55S0x/LPC550x 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 remain static.

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 "LPC55S0x/LPC550x 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 “LPC55S0x/LPC550x 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 CPU is 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 CPU state and registers, peripheral registers, and internal SRAM values are maintained, and the GPIO logic levels of the pins remain static.

GPIO pin interrupts, GPIO group interrupts, and selected peripherals such as SPI, I^{2}C, 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 “LPC55S0x/LPC550x 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 "LPC55S0x/LPC550x 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 "LPC55S0x/LPC550x 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 27 "LPC55S0x/LPC550x Real-Time Clock (RTC)". Interrupts must also be enabled via the power API.
- OS Event Timer. See Chapter 31 "LPC55S0x/LPC550x 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 “LPC55S0x/LPC550x 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 “LPC55S0x/LPC550x 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 27 “LPC55S0x/LPC550x Real-Time Clock (RTC)”. Interrupts must also be enabled via the power API.
- OS Event Timer. See Chapter 31 “LPC55S0x/LPC550x 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 "LPC55S0x/LPC550x 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 "LPC55S0x/LPC550x 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 "LPC55S0x/LPC550x 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 303. Register overview: pmc (base address = 0x40020000)

Name	Access	Offset	Description	Reset value	Section
STATUS[1]	RO	0x4	Power Management Controller FSM (Finite State Machines) status (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 [1]	RW	0x10	DCDC (first) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (DCDC 0).	0x010C 4E68	13.4.3
DCDC1 [1]	RW	0x14	DCDC (second) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (DCDC1).	0x0180 3A98	13.4.4
LDOPMU [1]	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).	0x69	13.4.6
REFFASTWKUP [1]	RW	0x40	Analog References fast wake-up Control register [Reset by: PoR].	0x1	13.4.7
XTAL32K [1]	RW	0x4C	32 KHz Crystal oscillator (XTAL) control register [Reset by: PoR, Brown Out Detectors Reset.	0x0020 4052	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 [1]	RW	0x90	Dummy Control bus to PMU [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset].	0x0	13.4.14
RTCOSC32K	RW	0x98	RTC clock control register (Reset by: PoR, Brown Out Detectors reset).	0x0	13.4.15
OSTIMER	RW	0x9C	OS timer control register (Reset by: PoR, Brown Out Detectors reset).	0x8	13.4.16
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

Table 303. Register overview: pmc (base address = 0x40020000) *…continued*

Name	Access	Offset	Description	Reset value	Section
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 [1]	W	0xD4	All SRAMs common control signals [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Software Reset].	0x1	13.4.20

[1] 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.

13.4.1 Status register[1]

This register shows Power Management Controller FSM status.

Table 304. Power Management Controller FSM (Finite State Machines) status (STATUS, offset = 0x4) bit description

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

[1] 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.

13.4.2 Reset control register

Table 305. Reset control (Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset) (RESETCTRL, offset = 0x8) bit description

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
		0	Reset event from deep-power down mode is disable.	
		1	Reset event from deep-power down mode is enable.	
2:1	-		Reserved. Only one should be written.	0x1
3	SWRRESETENABLE		Software reset enable.	0x0
		0	Software reset is disable.	
		1	Software reset is enable.	
5:4	BODVBATRES ETENA_SECU RE		BOD VBAT reset enable.	0x0
		0	BOD VBAT is disabled.	
		1	Any other value than b10, BOD VBAT reset is enabled.	
7:6	BODCORERES ETENA_SECU RE		BOD Core reset enable.	0x0
		0	BOD Core reset is disable.	
		1	Any other value than b10, BOD Core reset is enabled.	
27:8	-		Reserved. Only one should be written.	0x1
29:28	BODVBATRES ETENA_SECU RE_DP		BOD VBAT reset enable.	0x0
		0	BOD VBAT reset is disabled.	
		1	Any other value than b10, BOD VBAT reset is enabled.	
31:30	BODCORERES ETENA_SECU RE_DP		BOD Core reset enable.	0x0
		0	BOD Core reset is disabled.	
		1	Any other value than b10, BOD Core reset is enabled.	

13.4.3 DCDC first control register [1]

This shows DCDC (first) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset].

Table 306. DCDC (first) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (DCDC0), offset = 0x10) bit description

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.	0X09
16	DISABLEISENSE		Disable Current sensing.	0X0

Table 306. DCDC (first) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (DCDC0), offset = 0x10) *…continued*bit description

Bit	Symbol	Value	Description	Reset value
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	SLICINGENABLE		Enable staggered switching of power switches.	0X0
22	INDUCTORCLAMPENABLE		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.

[1] 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.

13.4.4 DCDC second control register [1]

This shows DCDC (second) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset].

Table 307. DCDC (second) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (DCDC1), offset = 0x14) bit description

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

Table 307. DCDC (second) control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (DCDC1), offset = 0x14) *…continued***bit description**

Bit	Symbol	Value	Description	Reset value
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

[1] 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.

13.4.5 Power Management Unit (PMU) and Always-On domains LDO control register [1]

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].

Table 308. 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, offset = 0x1C) bit description

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.	
		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

Table 308. 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, offset = 0x1C) *…continued*bit

Bit	Symbol	Value	Description	Reset value
20	BLEED		Controls LDOMEM bleed current.	0x0
		0b	Bleed current is disable	
		1b	Bleed current is enable	
23-21	-		Reserved Read value is undefined, only zero should be written.	undefined
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 .

[1] 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.

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 309. VBAT Brown Out Detector (BoD) control register (Reset by: PoR, pin reset, software reset) (BODVBAT, offset = 0x30) bit description

Bit	Symbol	Value	Description	Reset value
4:0	TRIGLVL		BoD trigger level.	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.	0x3
		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 [1]

This shows Analog References fast wake-up Control register [Reset by: PoR].

Table 310. Analog References fast wake-up Control register (REFFASTWKUP, offset = 0x40) bit description

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
		0	Analog References fast wake-up feature is disabled in case of Hardware Pin reset.	
		1	Analog References fast wake-up feature is enabled in case of Hardware Pin reset..	
1	HWWKUP		Analog References fast wake-up in case of Hardware Pin reset.	0x0
		0	Analog References fast wake-up feature is disabled in case of Hardware Pin reset.	
		1	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

[1] 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.

13.4.8 32 KHz Crystal oscillator (XTAL) control register [1]

This shows 32 KHz Crystal oscillator (XTAL) control register values [Reset by: PoR, Brown Out Detectors Reset].

Table 311. 32 KHz Crystal oscillator (XTAL) control register [Reset by: PoR, Brown Out Detectors Reset] (XTAL, offset = 0x4C) bit description

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
		0	Sourced from CAPTESTSTART.	
		1	Sourced from calibration.	
23	CAPTESTSTART		Start test.	0x0
24	CAPTESTENA BLE		Enable signal for cap test.	0x0

Table 311. 32 KHz Crystal oscillator (XTAL) control register [Reset by: PoR, Brown Out Detectors Reset] (XTAL, offset = 0x4C) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
25	CAPTESTOSCI NSEL		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.

[1] 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.

13.4.9 Analog comparator control register

Table 312. Analog comparator control register (Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset) (COMP, offset = 0x50) bit description

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.	
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.	

Table 312. Analog comparator control register (Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset) (COMP, offset = 0x50) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
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	
		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

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 313. 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
		0	Rising edge detection is disable.	
		1	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
		0	Falling edge detection is disable.	
		1	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
		0	Rising edge detection is disable.	
		1	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
		0	Falling edge detection is disable.	
		1	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
		0	Rising edge detection is disable.	
		1	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
		0	Falling edge detection is disable.	
		1	Falling edge detection is enable.	
6	RISINGEDGEWAKEUP3		Enable / disable detection of rising edge events on Wake Up 3 pin in Deep Power Down modes.	0x0
		0	Rising edge detection is disable.	
		1	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
		0	Falling edge detection is disable.	
		1	Falling edge detection is enable.	
11:8	-		Reserved. Read value is undefined, only zero should be written.	undefined
13:12	MODEWAKEUPIOPAD0		Selects function mode (on-chip pull-up/pull-down resistor control).	0x0
		0	Inactive. Inactive (no pull-down/pull-up resistor enabled.	
		1	Pull-down. Pull-down resistor enabled.	
		2	Pull-up. Pull-up resistor enabled.	

Table 313. Wake-up I/O Control register (WAKEUPIOCTRL, offset = 0x64) *…continued***bit description**

Bit	Symbol	Value	Description	Reset value
		3	Repeater. Repeater mode.	
15:14	MODEWAKEUPIOPAD1		Selects function mode (on-chip pull-up/pull-down resistor control).	0x0
		0	Inactive. Inactive (no pull-down/pull-up resistor enabled.	
		1	Pull-down. Pull-down resistor enabled.	
		2	Pull-up. Pull-up resistor enabled.	
		3	Repeater. Repeater mode.	
17:16	MODEWAKEUPIOPAD2		Selects function mode (on-chip pull-up/pull-down resistor control.	0x0
		0	Inactive. Inactive (no pull-down/pull-up resistor enabled.	
		1	Pull-down. Pull-down resistor enabled.	
		2	Pull-up. Pull-up resistor enabled.	
		3	Repeater. Repeater mode.	
19:18	MODEWAKEUPIOPAD3		Selects function mode (on-chip pull-up/pull-down resistor control.	0x0
		0	Inactive. Inactive (no pull-down/pull-up resistor enabled.	
		1	Pull-down. Pull-down resistor enabled.	
		2	Pull-up. Pull-up resistor enabled.	
		3	Repeater. Repeater mode.	
20	WAKEUPIO_ENABLE_CTRL		Enable WAKEUP IO PAD control from MODEWAKEUPIOPAD (bits 12 to 19.	0x0
		0	WAKEUP IO PAD mode control comes from IOCON.	
		1	WAKEUP IO PAD mode control comes from MODEWAKEUPIOPAD (bits 12 to 19.	
21	WAKEUPIO_RSTN		WAKEUP IO event detector reset control.	0x0
		0	Bloc is reset.	
		1	Bloc is not reset.	
31:22	-		Reserved. Read value is undefined, only zero should be written.	undefined

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 314. Wake-up I/O register (WAKEIOCAUSE, offset = 0x68) bit description

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.	

Table 314. Wake-up I/O register (WAKEIOCAUSE, offset = 0x68) *…continued*bit description

Bit	Symbol	Value	Description	Reset value
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 315. FRO and XTAL status register (Reset by: PoR, Brown Out Detectors reset) (STATUSCLK, offset = 0x74)bit description

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
13	RW	CDOGRESET[1]	The last chip reset was caused by the code Watchdog.	0
15:14	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.	-

[1] The values 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.

13.4.14 Dummy Control bus to PMU control register [1]

Dummy Control bus to PMU [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset].

Table 316. Dummy Control bus to PMU [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (MISCCTRL, offset = 0x90) bit description

Bit	Symbol	Value	Description	Reset value
0	LDODEEPSLE EPREF		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	LDOMEMHIGH ZMODE		Control the activation of LDO MEM High Z mode.	0x0
		0b	LDO MEM High Z mode is disabled.	
		1b	LDO MEM High Z mode is enabled.	
2	LOWPWR_FLA SH_BUF			0x0
11-3	MISCCTRL_3_ 11		Reserved.	0x0

Table 316. Dummy Control bus to PMU [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset] (MISCCTRL, offset = 0x90) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
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.	
15-13	MISCCTRL_13 _15		Reserved.	0x0
31-16	-		Reserved Read value is undefined, only zero should be written.	undefined.

[1] 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.

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 317. RTC 1 kHZ and 1 Hz clocks source control register (Reset by: PoR, Brown Out Detectors reset) (RTCOSC32K, offset = 0x98) bit description

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 318. OS timer control register [Reset by: PoR, Brown Out Detectors Reset] (OSTIMER, offset = 0x9C) bit description

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	
5:4	OSTIMERCLKSEL		OS event timer clock select.	0x0
		0	Oscillator 32 kHz clock	
		1	FRO 1 MHz clock	
		2	Main clock for OS timer.	
		3	No clock.	
31:6	-		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 319. Power configuration register 0 (PDRUNCFG0, offset = 0xB8) (Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		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 High Speed Crystal.	0x1
		0	Crystal 32 kHz is powered.	
		1	Crystal 32 kHz is powered-down.	

Table 319. Power configuration register 0 (PDRUNCFG0, offset = 0xB8) (Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
8	PDEN_XTAL32M		Controls power to High Speed Crystal.	0x1
		0	High Speed Crystal is powered.	
		1	High Speed Crystal 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.	
12:11	-		Reserved Only 'b11 should be written.	0x1
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.	0x3
17:16	-		Reserved. Only zero should be written.	0x2
18	-		Reserved Only 'b1 should be written.	0x1
19	PDEN_AUXBIAS		Controls power to auxiliary biasing (AUXBIAS)	0x1
		0	Auxiliary biasing is powered.	
		1	auxiliary biasing is powered-down.	
20	PDEN_LDOXO32M		Controls power to High Speed Crystal LDO.	0x1
		0	High Speed Crystal LDO is powered.	
		1	High Speed Crystal 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

13.4.18 Power configuration set register 0

The power configuration set register 0 controls the power to various analog blocks.

Table 320. Power configuration set register 0 (PDRUNCFGSET0, offset = 0xC0) (Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset) bit description

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 321. Power configuration clear register (PDRUNCFGCLR0, offset = 0xC8)(Reset by: PoR, pin reset, Brown Out Detectors reset, deep-power down reset, software reset) bit description

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 SRAM control register [1]

The power configuration clear register 0 controls the power to various analog blocks

Table 322. SRAM control register (SRAMCTRL, offset = 0xD4) bit description

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.

[1] 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.

UM11424

Chapter 14: LPC55S0x/LPC550x Power Profiles/Power Control API

Rev. 1.5 — 21 December 2023 **User manual**

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.
- Manages reset from BoD Vbat and BoD Core.

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.

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 323. 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. See Table 324 "DC-DC converter output voltage settings (default output is set to 1.10 V)".	14.4.3

By default, the internal DC-DC converter output voltage is set to 1.10 volts to accommodate frequencies of 96 MHz and below. However, the voltage can be modified to accommodate higher frequency ranges as described in Table 324 "DC-DC converter output voltage settings (default output is set to 1.10 V)".

Table 324. DC-DC converter output voltage settings (default output is set to 1.10 V)

Range	Frequency Ranges (MHz)	DC-DC Converter Output (V)
1	System Frequency ≤ 72	1.0 V - 1.1 V
2	System Frequency 73 to 96	1.025 V - 1.15 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.

Two Power APIs in the SDK power library are available to configure reset from BoD VBat and BoD Core.

Table 325. Power related API descriptions

Function prototype	API description	Section
`void POWER_SetBodVbatLevel (power_bod_vbat_level_tlevel,power_bod_hyst_t hyst, bool enBodVbatReset)`	Power API configures the threshold level and the hysteresis, and enable or disable the reset on low level detection on VBat.	
`void POWER_SetBodCoreLevel (power_bod_core_level_tlevel,power_bod_hyst_t hyst, bool enBodCoreReset)`	Power API configures the threshold level and the hysteresis, and enable or disable the reset on low level detection on Core.	

By default the Bod VBat is enable with a trigger level 1.65V and hysteresis of 75 mV. By default the Bod Core is enable with a trigger level 0.80V and hysteresis of 50 mV.

Remark:

- CPU 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_SetBodVbatLevel

This routine configures the device's internal BOD VBAT level. the API function configures the threshold level and the hysteresis, and enable or disable the reset on low level detection on Vbat.

Table 326. POWER_SetBodVbatLevel API routines

Routine	**POWER_SetVoltageForFreq**
SDK Prototype	`POWER_SetBodVbatLevel (power_bod_vbat_level_t level, power_bod_hyst_t hyst, bool enBodVbatReset);`
Input parameter	**Param0:** level **Param1:** hyst **Param2:** enBodVbatReset
Result	None.
Description	Configure BOD Vbat to reset the chip according to Vbat level.

14.4.1.1 Param0: level

The level parameter defines trig level used for the BoD VBat.

Table 327. Parameter level

Value	Description
0	Brown out detector VBAT level 1V.
1	Brown out detector VBAT level 1.1V.
2	Brown out detector VBAT level 1.2V.
3	Brown out detector VBAT level 1.3V.
4	Brown out detector VBAT level 1.4V.
5	Brown out detector VBAT level 1.5V.
6	Brown out detector VBAT level 1.6V.
7	Brown out detector VBAT level 1.65V.
8	Brown out detector VBAT level 1.7V.
9	Brown out detector VBAT level 1.75V.
10	Brown out detector VBAT level 1.8V.
11	Brown out detector VBAT level 1.9V.
12	Brown out detector VBAT level 2.0V.
13	Brown out detector VBAT level 2.1V.
14	Brown out detector VBAT level 2.2V.
15	Brown out detector VBAT level 2.3V.
16	Brown out detector VBAT level 2.4V.
17	Brown out detector VBAT level 2.5V.
18	Brown out detector VBAT level 2.6V.
19	Brown out detector VBAT level 2.7V.
20	Brown out detector VBAT level 2.8V.
21	Brown out detector VBAT level 2.9V.
22	Brown out detector VBAT level 3.0V.
23	Brown out detector VBAT level 3.1V.
24	Brown out detector VBAT level 3.2V.
25	Brown out detector VBAT level 3.3V.

14.4.1.2 Param1: hyst

The hyst parameter defines the hysteresis level used for the BoD VBat.

Table 328. Parameter hyst

Value	Description
0	BOD Hysteresis control level 25mV.
1	BOD Hysteresis control level 50mV
2	BOD Hysteresis control level 75mV
3	BOD Hysteresis control level 100mV

14.4.1.3 Param2: enBodVbatReset

The enBodVbatReset parameter defines the enable of the reset trigged by the BoD VBat.

Table 329. Parameter enBodVbatReset

Value	Description
0	The BoD VBat reset is disable.
1	The BoD VBat reset is enable.

14.4.2 POWER_SetBodCoreLevel

This routine configures the device's internal BOD Core level. the API function configures the threshold level and the hysteresis, and enable or disable the reset on low level detection on Vdd_core.

Table 330. POWER_SetBodCoreLevel API routines

Routine	POWER_SetVoltageForFreq
SDK Prototype	`POWER_SetBodCoreLevel (power_bod_core_level_t level, power_bod_hyst_t hyst, bool enBodCoreReset);`
Input parameter	**Param0:** level **Param1:** hyst **Param2:** enBodCoreReset
Result	None.
Description	Configure BOD Core to reset the chip according to Vdd_core level.

14.4.2.1 Param0: level

The level parameter defines trig level used for the BoD Core.

Table 331. Parameter level

Value	Description
0	Brown out detector Vdd_core level 0.60V.
1	Brown out detector Vdd_core level 0.65V.
2	Brown out detector Vdd_core level 0.70V.
3	Brown out detector Vdd_core level 0.75V.
4	Brown out detector Vdd_core level 0.80V.
5	Brown out detector Vdd_core level 0.85V.
6	Brown out detector Vdd_core level 0.90V.
7	Brown out detector Vdd_core level 0.95V.

14.4.2.2 Param1: hyst

The hyst parameter defines the hysteresis level used for the BoD Core.

Table 332. Parameter hyst

Value	Description
0	BOD Hysteresis control level 25mV.
1	BOD Hysteresis control level 50mV
2	BOD Hysteresis control level 75mV
3	BOD Hysteresis control level 100mV

14.4.2.3 Param2: enBodCoreReset

The enBodCoreReset parameter defines the enable of the reset trigged by the BoD Core.

Table 333. Parameter enBodCoreReset

Value	Description
0	The BoD Core reset is disable.
1	The BoD Core reset is enable.

14.4.3 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 334. 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.

14.4.3.1 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 96000000 inclusive.

14.4.4 POWER_GetWakeUpCause

This routine returns some key information related to the device reset wake-up cause, for all power modes.

Table 335. POWER_GetWakeUpCause API routine

Routine	CHIPLOWPOWER_enter_ sleep
SKD Prototype	`void POWER_GetWakeUpCause (power_reset_cause_t * reset_cause, power_boot_mode_t * boot_mode, power_wakeup_pin_t * wakeup_pin_cause)`
Input parameter	**Param0:** reset_cause **Param1:** boot_mode **Param2:** wakeup_pin_cause
Result	None
Description	The POWER_GetWakeUpCause API returns some key information related to the device reset wake-up cause, for all power modes.

14.4.4.1 Param0: reset_cause

The reset_cause parameter provides which reset source caused wake-up.

```
typedef enum _power_reset_cause

{

kRESET_CAUSE_POR = 0UL, /*!< Power On Reset */

kRESET_CAUSE_PADRESET = 1UL, /*!< Hardware Pin Reset */

kRESET_CAUSE_BODRESET = 2UL, /*!< Brown-out Detector reset (either
BODVBAT or BODCORE) */

kRESET_CAUSE_ARMSYSTEMRESET = 3UL, /*!< ARM System Reset */

kRESET_CAUSE_WDTRESET = 4UL, /*!< Watchdog Timer Reset */

kRESET_CAUSE_SWRRESET = 5UL, /*!< Software Reset */

kRESET_CAUSE_CDOGRESET = 6UL, /*!< Code Watchdog Reset */

/* Reset causes in DEEP-POWER-DOWN low power mode */

kRESET_CAUSE_DPDRESET_WAKEUPIO = 7UL, /*!< Any of the 5 wake-up pins */

kRESET_CAUSE_DPDRESET_RTC = 8UL, /*!< Real Time Clock (RTC) */

kRESET_CAUSE_DPDRESET_OSTIMER = 9UL, /*!< OS Event Timer (OSTIMER) */

kRESET_CAUSE_DPDRESET_WAKEUPIO_RTC = 10UL, /*!< Any of the 5 wake-up pins and

RTC (the 2 events occured within 1 nano-second of each other) */

kRESET_CAUSE_DPDRESET_WAKEUPIO_OSTIMER = 11UL, /*!< Any of the 5 wake-up pins and

OSTIMER (the 2 events occured within 1 nano-second of each other) */

kRESET_CAUSE_DPDRESET_RTC_OSTIMER = 12UL, /*!< Real Time Clock or OS Event Timer

(the 2 events occured within 1 nano-second of each other) */

kRESET_CAUSE_DPDRESET_WAKEUPIO_RTC_OSTIMER = 13UL, /*!< Any of the 5 wake-up pins or
     RTC

or OS Event Timer (the 3 events occured within 1 nano-second of each other) */

/* Miscallenous */

kRESET_CAUSE_NOT_RELEVANT = 14UL, /*!< No reset cause (for example,

this code is used when waking up from DEEP-SLEEP low power mode) */

kRESET_CAUSE_NOT_DETERMINISTIC = 15UL, /*!< Unknown Reset Cause. Should be

treated like "Hardware Pin Reset" from an application point of view. */

} power_reset_cause_t;
```

Note: In DEEP-POWER-DOWN mode, if several wake up sources have been enabled (for example, the RTC and a wake-up pin), the parameter contains the first event that occured. it is possible to know if the other enabled wake up event have occured by the reading directly PMC->AOREG1[DPDRESET_*] (DPDRESET_WAKEUPIO, DPDRESET_RTC and DPDRESET_OSTIMER).

14.4.4.2 Param1: boot_mode

The boot_mode parameter provides information on which low power mode or reset mode caused the wake-up to occur.

```
typedef enum _power_boot_mode
```

```
{
kBOOT_MODE_POWER_UP = 0UL, /*!< All non Low Power Mode wake up (Power On
Reset, Pin Reset, BoD Reset, ARM System Reset ... ) */
kBOOT_MODE_LP_DEEP_SLEEP = 1UL, /*!< Wake up from DEEP-SLEEP Low Power mode */
kBOOT_MODE_LP_POWER_DOWN = 2UL, /*!< Wake up from POWER-DOWN Low Power mode */
kBOOT_MODE_LP_DEEP_POWER_DOWN = 4UL, /*!< Wake up from DEEP-POWER-DOWN Low Power mode
     */
} power_boot_mode_t;
```

14.4.4.3 Param2: wakeup_pin_cause

This parameter provides information on which wake-pin caused the wake-up to occur.

```
typedef enum _power_wakeup_pin_t
{
kWAKEUP_PIN_NONE = 0UL, /*!< No wake up pin event */
kWAKEUP_PIN_0 = (1UL << 0), /*!< Wake up pin 0 event */
kWAKEUP_PIN_1 = (1UL << 1), /*!< Wake up pin 1 event */
kWAKEUP_PIN_2 = (1UL << 2), /*!< Wake up pin 2 event */
kWAKEUP_PIN_3 = (1UL << 3), /*!< Wake up pin 3 event */
kWAKEUP_PIN_4 = (1UL << 4), /*!< Wake up pin 4 event */
kWAKEUP_PIN_MULTIPLE = 0x1FUL, /*!< More than 1 wake up pins events occured (within
1 nano-second of each other) */
} power_wakeup_pin_t;
```

Note: If several wake up pins have been enabled (for example wakup pin 0, 3 and 4), the parameter contains the first wake up pin that triggered the device wake up. It is possible to know if the other enabled wake up pins event have occured by the reading directly PMC->WAKEIOCAUSE.

14.4.4.4 Examples

Following is an example of using the routines:

```
int main()
{
power_reset_cause_t reset_cause;
power_boot_mode_t boot_mode;
power_wakeup_pin_t wakeup_pin_cause;
```

```
if ( POWER_PowerInit() != kPOWER_Status_Success )

{

/* On failure, stop processing */

}

/* On success, continue normal processing ... */

/* Get device wake up cause */

POWER_GetWakeUpCause(&reset_cause, &boot_mode, &wakeup_pin_cause);

if ( boot_mode == kBOOT_MODE_POWER_UP )

{ /* All wake up from non low power mode */

if ( reset_cause == kRESET_CAUSE_POR )

{ /* Power On Reset */

/* ... */

}
```

14.4.5 POWER_EnterSleep

This routine puts the device in sleep mode.

Table 336. 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 */
}
```

14.4.6 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 337. 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.

14.4.6.1 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 338. 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

Table 338. Parameter exclude_from_pd *…continued*

Bit	Symbol	Description	Value
11	-	Reserved	-
12	-	Reserved	-
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	-	Reserved	-
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
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	-

14.4.6.2 Param1: sram_retention_ctrl

The sram_retention_ctrl parameter defines which SRAM instances are 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 in the table 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 339. Parameter sram_retention_ctrl

Bit	SRAM instance	Value
0	**SRAM_X0** (4 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
1	**SRAM_X1** (4 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
2	**SRAM_X2** (4 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
3	**SRAM_X3** (4 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
4	**SRAM_00** (32 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
5	Reserved	0: bit reserved. 1: bit reserved.
6	**SRAM_1** (16 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
7	**SRAM_2** (16kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
13:8	Reserved	0: bit reserved. 1: bit reserved.
14	**SRAM3** (16 kBytes)	0: SRAM keeps current state during deep-sleep. 1: SRAM in retention mode during deep-sleep.
31:15	Reserved	-

14.4.6.3 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 340. 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

Table 340. Parameter wakeup_interrupts *…continued*

Bit	Wake-up source	Description	Deep-sleep	Power-down	Deep power-down
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
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	-	-	-	-	-
28	-	-	-	-	-
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 enable 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	-	-	-	-	-
43	CAN0_INT0	CAN	NO	NO	NO
44	CAN0_INT1	CAN	NO	NO	NO
45	-	-	-	-	-
46	-	-	-	-	-
47	-	-	-	-	-

Table 340. Parameter wakeup_interrupts *…continued*

Bit	Wake-up source	Description	Deep-sleep	Power-down	Deep power-down
48	-	-	-	-	-
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
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	-	-	-	-	-
58	WAKEUP_SDMA1	Secure system DMA	YES	NO	NO
59	WAKEUP_HS_SPI	high-speed SPI	YES	NO	NO
60	CDOG	Code watchdog	NO	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.

14.4.6.4 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 341. 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

Table 341. Parameter hardware_wake_ctrl

Bit	Symbol	Description	Value
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
31:6	Reserved	Should always be set to zero.	Shall always be set to “0”

14.4.7 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 342. 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*.

14.4.7.1 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 343. Parameter exclude_from_pd

Bit	Symbol	Description	Value
0	-	Reserved	-
1	BIAS	Analog references	0: Powered down 1: Running
5:2	-	Reserved	-

Table 343. Parameter exclude_from_pd ...continued

Bit	Symbol	Description	Value
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
21:14	-	Reserved	-
22	RNG	True random number generator.	0: Powered down 1: Entropy is saved
31:23	-	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 memories regulator (LDOMEM) remains in running mode for retention purposes.

The analog references (BIAS) are required only when the analog comparator is a wake-up source.

14.4.7.2 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_2000 - 0x0400_25FF] inside RAMX_2 is used to store the state of CPU (which means that any user data in this area prior to calling the low power API will be lost). Therefore, RAM_X2 retention mode is always 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 344. Parameter sram_retention_ctrl

Bit	SRAM instance	Value
0	**SRAM_X0** (4 kBytes)	0: SRAM keeps current state during power-down. 1: SRAM in retention mode during power-down.
1	**SRAM_X1** (4 kBytes)	0: SRAM keeps current state during power-down. 1: SRAM in retention mode during power-down.
2	**SRAM_X2** (4 kBytes)	The RAM_X2 is always in retention mode for CPU retention.
3	**SRAM_X3** (4 kBytes)	0: SRAM keeps current state during power-down. 1: SRAM in retention mode during power-down.

Table 344. Parameter sram_retention_ctrl *…continued*

Bit	SRAM instance	Value
4	**SRAM_00** (16 kBytes)	0: SRAM keeps current state during power-down.
5	**SRAM_01** (16 kBytes)	0: SRAM keeps current state during power-down. 1: SRAM in retention mode during power-down.
6	**SRAM_1** (16 kBytes)	0: SRAM keeps current state during power-down. 1: SRAM in retention mode during power-down.
7	**SRAM_2** (16 kBytes)	0: SRAM keeps current state during power-down. 1: SRAM in retention mode during power-down.
13:8	Reserved	0: bit reserved. 1: bit reserved.
14	**SRAM3** (16 kBytes)	0: SRAM keeps current state during power-down. 1: SRAM in retention mode during power-down.
31:15	Reserved	Only write 0x0

14.4.7.3 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.

14.4.7.4 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 14.4.6.2 “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_2000 - 0x0400_25FF] 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 14.4.6.2 “Param1: sram_retention_ctrl”.

The cpu_retention_ctrl parameter is a 32-bit value defined below:

Table 345. 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.

14.4.8 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 346. 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.

14.4.8.1 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 347. 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.

14.4.8.2 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_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 64 KB).

Table 348. Parameter sram_retention_ctrl

Bit	SRAM instance	Value
0	**SRAM_X0** (4 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
1	**SRAM_X1** (4 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
2	**SRAM_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	**SRAM_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.
5:4	**Reserved**	Only write 0x0
6	**SRAM_1** (16 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
7	**SRAM_2** (16 kBytes)	0: SRAM in shutdown mode during power-down/deep power-down. 1: SRAM in retention mode during power-down/deep power-down.
13:8	Reserved	Only write 0x0
14	**SRAM3** (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

14.4.8.3 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.

14.4.8.4 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 349 shows wake_up_io_ctrl parameter is a 32-bit value.

Table 349. 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
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

Table 349. Parameter wakeup_io_ctrl

Bit	Symbol	Description	Value
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	USEEXTERNALPULLUPDOWNWAKEUP0	Enable usage of the external pull-up/pull-down for Wake-up pin 0.	0: Use internal Pull-up/Pull-down. 1: Use external Pull-up/Pull-down.
17	USEEXTERNALPULLUPDOWNWAKEUP1	Enable usage of the external pull-up/pull-down for Wake-up pin 1.	0: Use internal Pull-up/Pull-down. 1: Use external Pull-up/Pull-down.
18	USEEXTERNALPULLUPDOWNWAKEUP2	Enable usage of the external pull-up/pull-down for Wake-up pin 2.	0: Use internal Pull-up/Pull-down. 1: Use external Pull-up/Pull-down.
19	USEEXTERNALPULLUPDOWNWAKEUP3	Enable usage of the external pull-up/pull-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 = 0x40FF; /* 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 = 0x40FF & (~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 = 0x40FF; /* 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);
```

14.5.3 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 ; /* */
```

14.5.3.1 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);
```

14.5.3.2 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);
```

14.5.3.3 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_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_PIO0_PULLUPDOWN_INDEX 8  /*!< Wake-up I/O 0 pull-up/down
      configuration index */

#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 */

#define LOWPOWER_WAKEUPIO_PIO0_USEEXTERNALPULLUPDOWN_INDEX 16 /*!< Wake-up I/O 0 use
     external pull-up/down disable/enable control index */

#define LOWPOWER_WAKEUPIO_PIO1_USEEXTERNALPULLUPDOWN_INDEX 17 /*!< Wake-up I/O 1 use
     external pull-up/down disable/enable control index */

#define LOWPOWER_WAKEUPIO_PIO2_USEEXTERNALPULLUPDOWN_INDEX 18 /*!< Wake-up I/O 2 use
     external pull-up/down disable/enable control index */

#define LOWPOWER_WAKEUPIO_PIO3_USEEXTERNALPULLUPDOWN_INDEX 19 /*!< Wake-up I/O 3 use
     external pull-up/down disable/enable control index */

#define LOWPOWER_WAKEUPIO_PIO0_USEEXTERNALPULLUPDOWN_MASK (1UL <<
     LOWPOWER_WAKEUPIO_PIO0_USEEXTERNALPULLUPDOWN_INDEX) /*!< Wake-up I/O 0 use
     external pull-up/down disable/enable mask, 0: disable, 1: enable */

#define LOWPOWER_WAKEUPIO_PIO1_USEEXTERNALPULLUPDOWN_MASK (1UL <<
     LOWPOWER_WAKEUPIO_PIO1_USEEXTERNALPULLUPDOWN_INDEX) /*!< Wake-up I/O 1 use
     external pull-up/down disable/enable mask, 0: disable, 1: enable */

#define LOWPOWER_WAKEUPIO_PIO2_USEEXTERNALPULLUPDOWN_MASK (1UL <<
     LOWPOWER_WAKEUPIO_PIO2_USEEXTERNALPULLUPDOWN_INDEX) /*!< Wake-up I/O 2 use
     external pull-up/down disable/enable mask, 0: disable, 1: enable */
```

```
#define LOWPOWER_WAKEUPIO_PIO3_USEEXTERNALPULLUPDOWN_MASK (1UL <<
     LOWPOWER_WAKEUPIO_PIO3_USEEXTERNALPULLUPDOWN_INDEX) /*!< Wake-up I/O 3 use
     external pull-up/down disable/enable mask, 0: disable, 1: enable */

*/

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);
```

UM11424

Chapter 15: LPC55S0x/LPC550x I/O Pin Configuration (IOCON)

Rev. 1.5 — 21 December 2023 **User manual**

15.1 How to read this chapter

The IOCON block is included on all LPC55S0x/LPC550x 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 “LPC55S0x/LPC550x 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^{2}C/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^{2}C-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 of 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 44 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 45 shows the control of a combo I^{2}C/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.

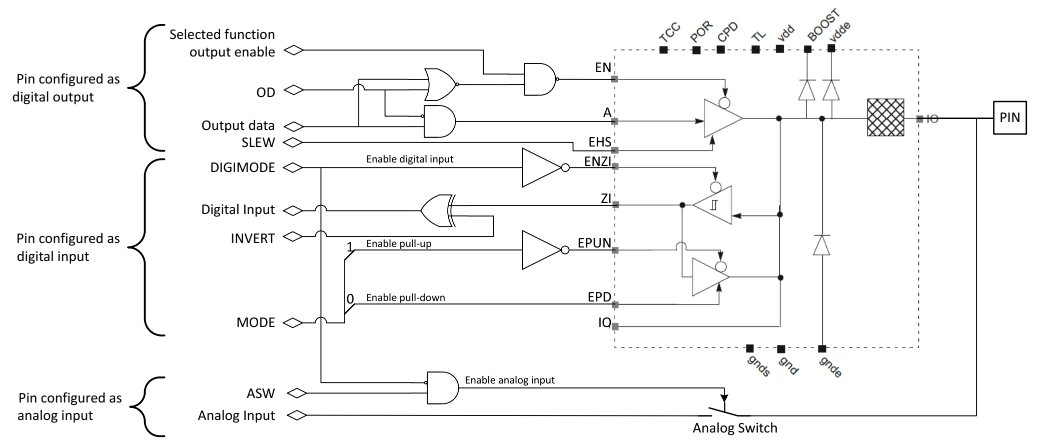


Fig 44. Standard GPIO pin configuration

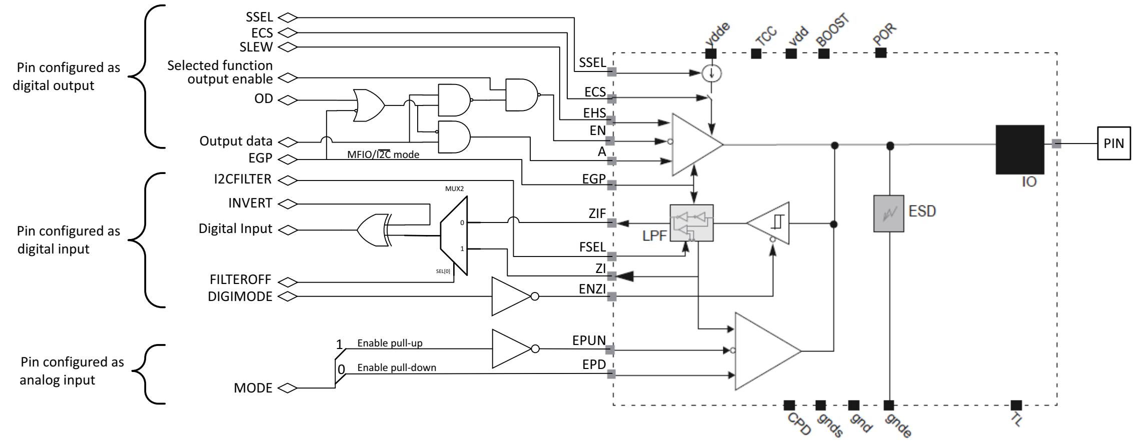


Fig 45. Combo I^{2}C/GPIO pin configuration

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^{2}C/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^{2}C mod, depending on the value of I^{2}CFILTER 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^{2}C modes

Pins that support I^{2}C with specialized pad electronics (PIO0_13 and PIO0_14) have additional configuration bits. These have multiple configurations to support I^{2}C variants. These are not hard-wired so that the pins can be easily used for non-I^{2}C functions. See Table 353 for recommended mode settings.

For non-I^{2}C 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^{2}C/MFIO pins when I^{2}C 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.

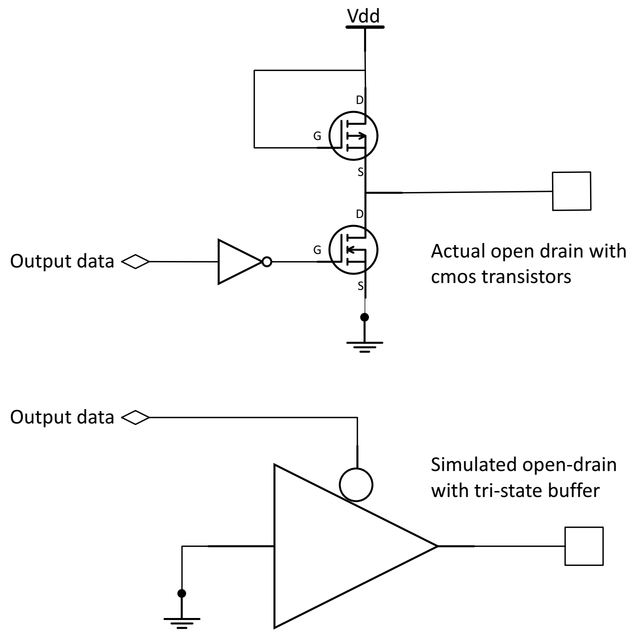

Vdd
D
G
S
D
G
S
Output data
Actual open drain with cmos transistors
Output data
Simulated open-drain with tri-state buffer

Fig 46. 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 350 exist on each package configuration.

One may identify three pin types:

- Digital only pin (D).
- Analog/digital pins (A).
- I^{2}C pin (I).

Table 350 gives an overview of IOCON registers. All of them are 32-bit RW registers. Some bit fields are not used and are reserved.

Table 350. 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	Reserved.	-	-		
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 - 0x094	Digital I/O control for port 1 pins PIO1_b (PIO1_1 - PIO1_6).	RW	D	0x0000	15.5.1
0x0A4	Digital I/O control for port 1 pins PIO1_b (PIO1_9).	RW	A	0x0000	15.5.3
0x0A8 - 0x0AC	Digital I/O control for port 1 pins PIO1_b (PIO1_10 - PIO1_11).	RW	D	0x0000	15.5.1
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 351 applies to pins referenced as pin type D in Table 350.

Reset values concern all pin of this type except PIO0_2 and PIO0_5 (see notes [1] and [2]).

Table 351. Type D IOCON registers

Bit	Symbol	Value	Description	Reset value
3:0	FUNC	-	Selects pin function. See Table 357, Table 358, and Table 359.	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 352 applies to pins PIO0_13 and PIO0_14. See Table 353 for recommended setting for I^{2}C operation.

Table 352. Type I IOCON registers

Bit	Symbol	Value	Description	Reset value
3:0	FUNC	-	Selects pin function. See Table 357, Table 358, and Table 359.	0
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.	

Table 352. Type I IOCON registers *…continued*

Bit	Symbol	Value	Description	Reset value
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^{2}C 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^{2}C mode.	
		1	1V8 signaling in I^{2}C 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^{2}C mode (EGP = 0), noise pulses below approximately 10 ns or 50 ns are filtered out, depending on I^{2}CFILTER bit field value.	
		1	Filter disabled. No input filtering is done.	
13	ECS		Pull-up current source enable in I^{2}C 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^{2}C mode.	1
		0	I^{2}C mode.	
		1	GPIO mode.	
15	I2CFILTER		Configures I^{2}C features for Standard-mode, Fast-mode, Fast-mode Plus operation and High-Speed mode operation	0
		0	I^{2}C 50 ns glitch filter enabled. Typically used for Standard mode, Fast-mode and Fast-mode Plus I^{2}C.	
		1	I^{2}C 10 ns glitch filter enabled. Typically used for High-Speed mode I^{2}C.	
31:16	-	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 353. Suggested IOCON settings for I^{2}C functions

Mode	IOCON register bit						
	15: I^{2}CFILTER	14: I^{2}CDRIVE	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 I^{2}C	1	0	0	0	1	0	0
Fast-mode Plus I^{2}C	0	0	0	0	1	0	0
High-speed slave I^{2}C	-	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 354 applies to pins referenced as pin type A in Table 350.

Reset values all pins of this type except PIO0_11 and PIO0_12 (see notes [1] and [2]).

Table 354. Type A IOCON registers

Bit	Symbol	Value	Description	Reset value
3:0	FUNC	-	Selects pin function. See Table 357, Table 358, and Table 359.	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).	
10	ASW		Analog switch input control. Usable only if DIGIMODE = 0.	0
		0	Analog switch is open (disabled).	
		1	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).

[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 set the ASW bit to '1' in the related IOCON register.

In GPIO module, set the related GPIO bit direction (DIR) as input. Table 355 shows the analog input related to each pin of type A and Table 356 gives the pin associated to each analog input function.

Table 355. 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	-	-
PIO0_11	ADC0_9	-	-
PIO0_12	ADC0_10	-	-
PIO0_15	ADC0_2	-	-
PIO0_16	ADC0_8	-	-
PIO0_18	ACMP0_C	-	-
PIO0_23	ADC0_0	-	-
PIO0_31	ADC0_3	-	-
PIO1_0	ADC0_11	-	-
PIO1_8	ADC0_4	-	-
PIO1_9	ADC0_12	-	-

Table 356. 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		
ADC0_8	P0_16		
ADC0_9	P0_11		
ADC0_10	P0_12		
ADC0_11	P1_0		
ADC0_12	P1_9		

The FUNC field for PIO0_11 and PIO0_12 resets to 0b110 (0x6), selecting the Serial Wire Debug function by default (SWCLK).

15.5.4 IOCON pin functions in relation to FUNC values

Table 357, Table 358, and Table 359 show the functions associated to each pin. FUNC value controls the function that is connected to the pin.

Table 357. 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	CTIMER0_MAT0	SCT0_GPI0
P0_1	-	FC3_CTS_SDA_SSEL0	CTIMER_INP0	SCT0_GPI1
P0_2	FC3_TXD_SCL_MISO_WS	CTIMER_INP1	SCT0_OUT0	SCT0_GPI2
P0_3	FC3_RXD_SDA_MOSI-DATA	CTIMER_MAT1	SCT0_OUT1	SCT0_GPI3
P0_4	CAN0_RD	FC4_SCK	CTIMER_INP12	SCT0_GPI4
P0_5	CAN0_TD	FC4_RXD_SDA_MOSI_DATA	CTIMER3_MAT0	SCT0_GPI5
P0_6	FC3_SCK	CTIMER_INP13	CTIMER4_MAT0	SCT0_GPI6
P0_7	FC3_RTS_SCL_SSEL1	-	FC5_SCK	FC1_SCK
P0_8	FC3_SSEL3	-	FC5_RXD_SDA_MOSI_DATA	SWO
P0_9	FC3_SSEL2	-	FC5_TXD_SCL_MISO_WS	-
P0_10	FC6_SCK	CTIMER_INP10	CTIMER2_MAT0	FC1_TXD_SCL_MISO_WS
P0_11	FC6_RXD_SDA_MOSI_DATA	CTIMER2_MAT2	FREQME_GPIO_CLK_A	-
P0_12	FC3_TXD_SCL_MISO_WS	-	FREQME_GPIO_CLK_B	SCT0_GPI7
P0_13	FC1_CTS_SDA_SSEL0	UTICK_CAP0	CTIMER_INP0	SCT0_GPI0
P0_14	FC1_RTS_SCL_SSEL1	UTICK_CAP1	CTIMER_INP1	SCT0_GPI1
P0_15	FC6_CTS_SDA_SSEL0	UTICK_CAP2	CTIMER_INP16	SCT0_OUT2
P0_16	FC4_TXD_SCL_MISO_WS	CLKOUT	CTIMER_INP4	-
P0_18	FC4_CTS_SDA_SSEL0	-	CTIMER1_MAT0	SCT0_OUT1
P0_19	FC4_RTS_SCL_SSEL1	UTICK_CAP0	CTIMER0_MAT2	SCT0_OUT2
P0_20	FC3_CTS_SDA_SSEL0	CTIMER1_MAT1	CTIMER_INP15	SCT0_GPI2
P0_21	FC3_RTS_SCL_SSEL1	UTICK_CAP3	CTIMER3_MAT3	SCT0_GPI3
P0_22	FC6_TXD_SCL_MISO	UTICK_CAP1	CTIMER_INP15	SCT0_OUT3
P0_23	MCLK	CTIMER1_MAT2	CTIMER3_MAT3	SCT0_OUT4
P0_24	FC0_RXD_SDA_MOSI_DATA	-	CTIMER_INP8	SCT0_GPI0
P0_25	FC0_TXD_SCL_MISO_WS	-	CTIMER_INP9	SCT0_GPI1
P0_26	FC2_RXD_SDA_MOSI_DATA	CLKOUT	CTIMER_INP14	SCT0_OUT5
P0_27	FC2_TXD_SCL_MISO_WS	-	CTIMER3_MAT2	SCT0_OUT6
P0_28	FC0_SCK	-	CTIMER_INP11	SCT0_OUT7
P0_29	FC0_RXD_SDA_MOSI_DATA	-	CTIMER2_MAT3	SCT0_OUT8
P0_30	FC0_TXD_SCL_MISO_WS	-	CTIMER0_MAT0	SCT0_OUT9
P0_31	FC0_CTS_SDA_SSEL0	-	CTIMER0_MAT1	SCT0_OUT3

Table 357. 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_0	FC0_RTS_SCL_SSEL1	-	CTIMER_INP2	SCT0_GPI4
P1_1	FC3_RXD_SDA_MOSI_DATA	-	CTIMER_INP3	SCT0_GPI5
P1_2	CAN0_TD	-	CTIMER0_MAT3	SCT0_GPI6
P1_3	CAN0_RD	-	-	SCT0_OUT4
P1_4	FC0_SCK	-	CTIMER2_MAT1	SCT0_OUT0
P1_5	FC0_RXD_SDA_MOSI_DATA	-	CTIMER2_MAT0	SCT0_GPI0
P1_9	-	FC1_SCK	CTIMER_INP4	SCT0_OUT2
P1_10	-	FC1_RXD_SDA_MOSI_DATA	CTIMER1_MAT0	SCT0_OUT3
P1_11	-	FC1_TXD_SCL_MISO_WS	CTIMER_INP5	-
P1_21	FC7_CTS_SDA_SSEL0	-	CTIMER3_MAT2	-
P1_22	-	-	CTIMER2_MAT3	SCT0_GPI5
P1_23	FC2_SCK	SCT0_OUT0	-	-
P1_25	FC2_TXD_SCL_MISO_WS	SCT0_OUT2	-	UTICK_CAP0
P1_29	FC7_RXD_SDA_MOSI_DATA	-	SCT0_GPI6	-

Table 358. 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	-	-	-	-	-
P0_1	-	-	CMP0_OUT	-	-
P0_2	-	-	-	-	-
P0_3	-	-	-	-	-
P0_4	-	-	-	FC3_CTS_SDA_SSEL0	-
P0_5	-	-	-	FC3_RTS_SCL_SSEL1	MCLK
P0_6	-	-	-	-	-
P0_7	-	-	-	-	-
P0_8	-	-		-	-
P0_9	-	-	-	-	-
P0_10	SCT0_OUT2	SWO	-	-	-
P0_11	-	SWCLK	-	-	-
P0_12	-	SWDIO	FC6_TXD_SCL_MISO_WS	-	-
P0_13	FC1_RXD_SDA_MOSI_DATA	-	-	-	PLU_INPUT0
P0_14	-	FC1_TXD_SCL_MISO_WS	-	-	PLU_INPUT1
P0_15	-	-	-	-	-
P0_16	-	-	-	-	-

Table 358. 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_18	-	-	-	-	PLU_INPUT3
P0_19		-	FC7_TXD_SCL_MISO_WS	-	PLU_INPUT4
P0_20	-	-	FC7_RXD_SDA_MOSI_DATA	HS_SPI_SSEL0	PLU_INPUT5
P0_21	-	-	FC7_SCK	HS_SPI_SSEL3	PLU_CLKIN
P0_22	-	-	-	-	PLU_OUT7
P0_23	FC0_CTS_SDAX_SSEL0	-	-	-	-
P0_24	-	-	-	-	-
P0_25	-	-	-	-	-
P0_26	-	-	-	FC0_SCK	HS_SPI_MOSI
P0_27	-	-	FC7_RXD_SDA_MOSI_DATA	-	PLU_OUT0
P0_28	-	-	-	-	PLU_OUT1
P0_29	-	-	CMP0_OUT	-	PLU_OUT2
P0_30	-	-		-	-
P0_31	-	-	-	-	-
P1_0	-	-	-	-	PLU_OUT3
P1_1	HS_SPI_SSEL1	-	-	-	PLU_OUT4
P1_2	-	HS_SPI_SCK	-	-	PLU_OUT5
P1_3	-	HS_SPI_MISO	-	-	PLU_OUT6
P1_4	FREQME_GPIO_CLK_A	-	-	-	-
P1_5	-	-	-	-	-
P1_9	FC4_CTS_SDA_SSEL0	-	-	-	-
P1_10	-	-	-	-	-
P1_11	-	-	-	-	-
P1_21	FC4_RXD_SDA_MOSI_DATA		PLU_OUT3	-	-
P1_22	FC4_SSEL3	-	PLU_OUT4	-	CAN0_RD
P1_23	FC3_SSEL2	-	PLU_OUT5	-	-
P1_25	-	-	PLU_CLKIN	-	-
P1_29	-	-	PLU_INPUT2	-	-

Table 359. I/O control registers: FUNC values (FUNC = 10 to 15) and pin functions

Pin	FUNC = 10	FUNC = 11
P0_0	SEC_PIO0_0	-
P0_1	SEC_PIO0_1	-
P0_2	SEC_PIO0_2	-
P0_3	SEC_PIO0_3	-
P0_4	SEC_PIO0_4	-
P0_5	SEC_PIO0_5	-

Table 359. I/O control registers: FUNC values (FUNC = 10 to 15) and pin functions *…continued*

Pin	FUNC = 10	FUNC = 11
P0_6	SEC_PIO0_6	-
P0_7	SEC_PIO0_7	-
P0_8	SEC_PIO0_8	-
P0_9	SEC_PIO0_9	-
P0_10	SEC_PIO0_10	-
P0_11	SEC_PIO0_11	-
P0_12	SEC_PIO0_12	-
P0_13	SEC_PIO0_13	-
P0_14	SEC_PIO0_14	-
P0_15	SEC_PIO0_15	-
P0_16	SEC_PIO0_16	-
P0_18	SEC_PIO0_18	-
P0_19	SEC_PIO0_19	-
P0_20	SEC_PIO0_20	FC4_TXD_SCL_MISO_WS
P0_21	SEC_PIO0_21	-
P0_22	SEC_PIO0_22	-
P0_23	SEC_PIO0_23	-
P0_24	SEC_PIO0_24	-
P0_25	SEC_PIO0_25	-
P0_26	SEC_PIO0_26	-
P0_27	SEC_PIO0_27	-
P0_28	SEC_PIO0_28	-
P0_29	SEC_PIO0_29	-
P0_30	SEC_PIO0_30	-
P0_31	SEC_PIO0_31	-

UM11424

Chapter 16: LPC55S0x/LPC550x General Purpose I/O (GPIO)

Rev. 1.5 — 21 December 2023 **User manual**

16.1 How to read this chapter

The 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 the GPIO registers are reserved. See Table 366.

Table 360. GPIO pins available

Package	Total GPIOs	GPIO Port 0	GPIO Port 1
HTQFP64	45	PIO0_0 to PIO0_16, PIO0_18 to PIO0_31	PIO1_0 to PIO1_5, PIO1_9 to PIO0_11, PIO1_21 to PIO1_23, PIO1_25, PIO1_29
HVQFN48	30	PIO0_2 to PIO0_6, PIO0_8 to PIO0_16, PIO0_20 to PIO0_23, PIO0_25 to PIO0_27, PIO0_29 to PIO0_30	PIO1_0 to PIO1_5, PIO1_21

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 43.3.4 “Interrupt, DMA and GPIO: Secure instance and masking”.

16.3 Basic configuration

For the GPIO port registers, enable the clock for 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 “LPC55S0x/LPC550x Pin Interrupt and Pattern Match (PINT)” and Chapter 21 “LPC55S0x/LPC550x: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 that is configured as input or set the state of each pin that 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 361. 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 362. 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. One register for each port pin. Supported pins depends on the specific device and package.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 363. 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. One register for each port pin. Supported pins depends on the specific device and package. **Remark:** One register for each port pin. Supported pins depends 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 364. 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 365. 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 366. 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 367. 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 368. 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 369. 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 370. 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 371. 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 372. 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 toggled by writing ones to these write-only registers.

Table 373. 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 43.3.4 “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.

UM11424

Chapter 17: LPC55S0x/LPC550x Secure General Purpose I/O (Secure GPIO)

Rev. 1.5 — 21 December 2023 User manual

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 "LPC55S0x/LPC550x Secure pin interrupt and pattern match (Secure PINT)" and Chapter 18 "LPC55S0x/LPC550x 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 "LPC55S0x/LPC550x 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 374. 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 375. 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). One register for each pin. 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 376. 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 377. 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 378. 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 379. 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 driven to the pins.

Table 380. 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 381. 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 382. 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 383. 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 384. 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 385. 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 386. 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), and
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 several 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 that are being 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 a 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 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.

UM11424

Chapter 18: LPC55S0x/LPC550x Input Multiplexing (INPUTMUX)

Rev. 1.5 — 21 December 2023 **User manual**

18.1 How to read this chapter

Input multiplexing is present on all LPC55S0x/LPC550x 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 "LPC55S0x/LPC550x 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 4.5.16 "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 387. 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 23 "LPC55S0x/LPC550x 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 47 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.

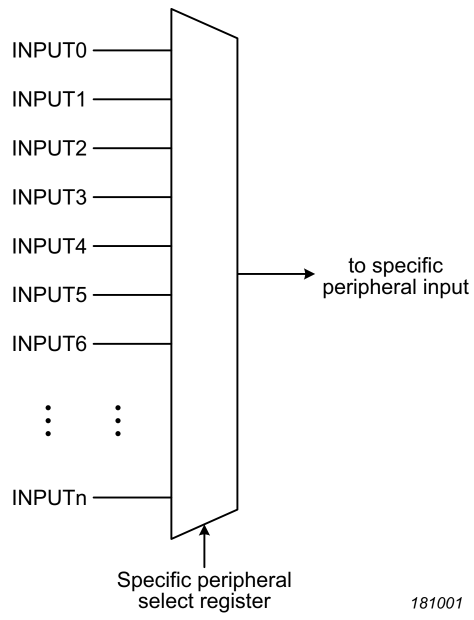

INPUT0
INPUT1
INPUT2
INPUT3
INPUT4
INPUT5
INPUT6
INPUTn
to specific peripheral input
Specific peripheral select register
181001

Fig 47. 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 48 shows the detail of this multiplexing.

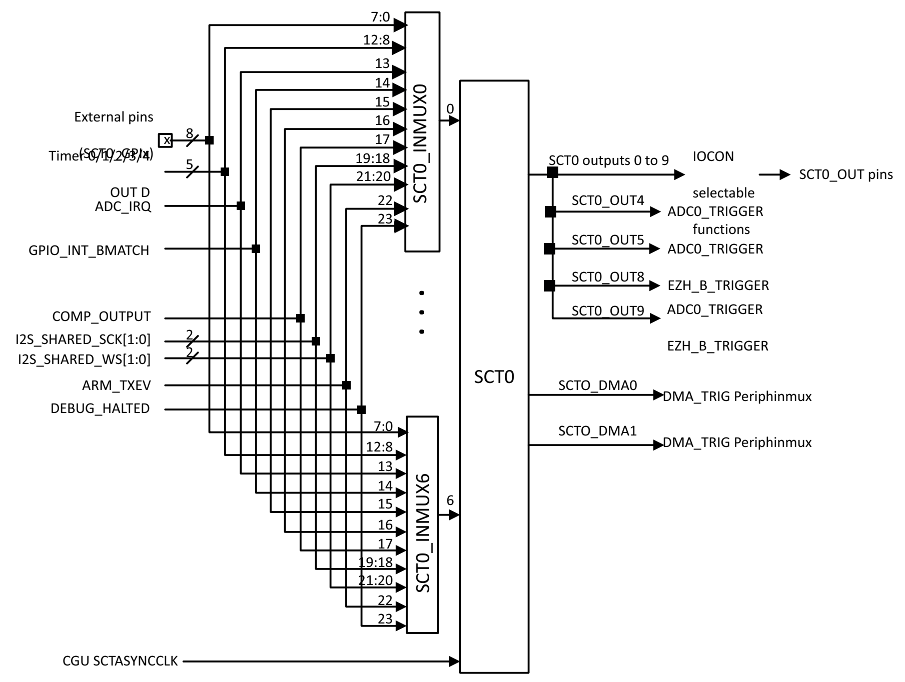


Fig 48. SCT0 input multiplexing

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.

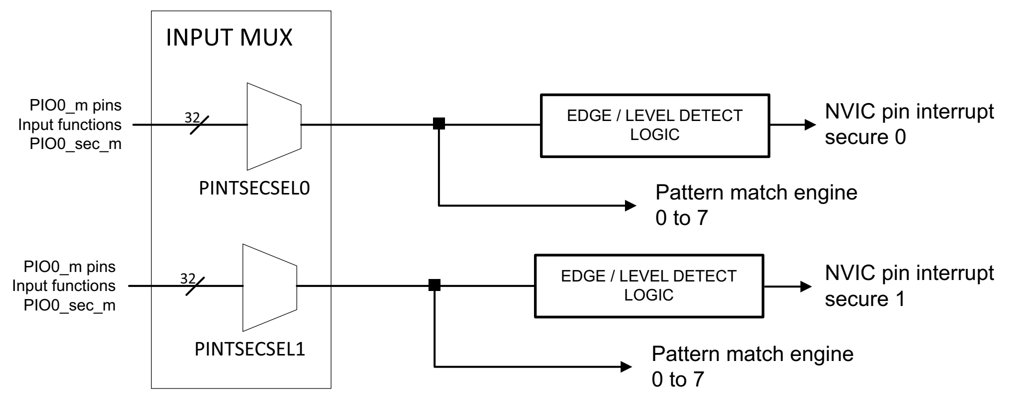

INPUT MUX
PIO0_m pins
Input functions
PIO0_sec_m
32
PINTSECSEL0
EDGE / LEVEL DETECT LOGIC
NVIC pin interrupt secure 0
Pattern match engine 0 to 7
PIO0_m pins
Input functions
PIO0_sec_m
32
PINTSECSEL1
EDGE / LEVEL DETECT LOGIC
NVIC pin interrupt secure 1
Pattern match engine 0 to 7

Fig 49. 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).

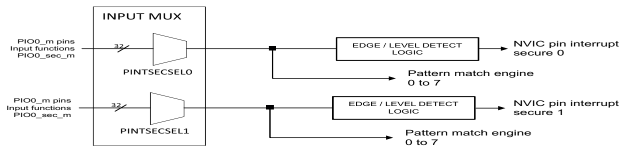

INPUT MUX
PIO0_m pins
Input functions
PIO0_sec_m
32
PINTSECSEL0
EDGE / LEVEL DETECT LOGIC
NVIC pin interrupt secure 0
Pattern match engine 0 to 7
PIO0_m pins
Input functions
PIO0_sec_m
32
PINTSECSEL1
EDGE / LEVEL DETECT LOGIC
NVIC pin interrupt secure 1
Pattern match engine 0 to 7

Fig 50. 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 51. 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".

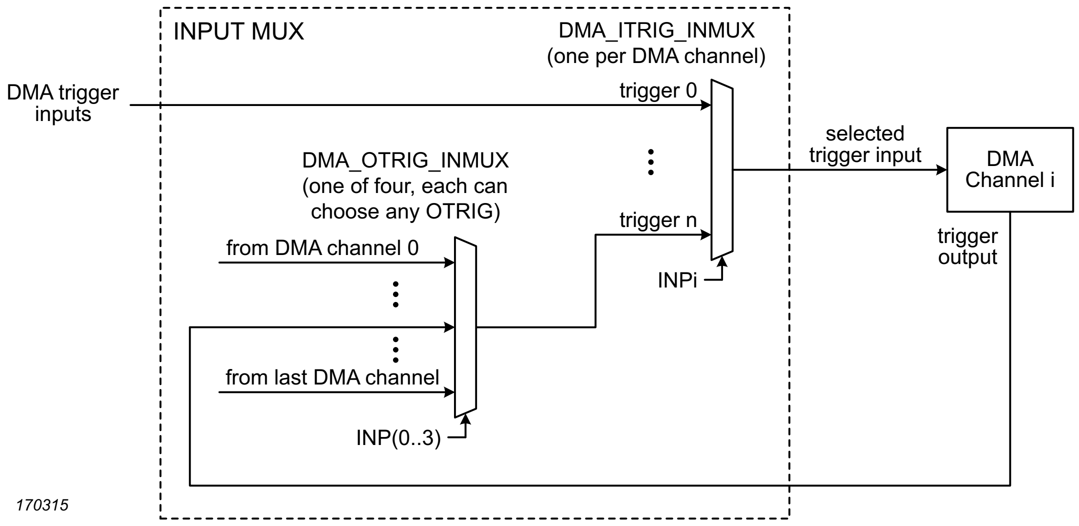


Fig 51. DMA trigger input multiplexing

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 388 are reserved and should not be written.

Table 388. 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 388. 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 388. 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 389. 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	Reserved.	
			0x10	Reserved.	
			0x11	COMP_OUTPUT output from analog comparator.	
			0x12	I2S_SHARED_SCK[0] output from I^{2}S pin sharing.	
			0x13	I2S_SHARED_SCK[1] output from I^{2}S pin sharing.	
			0x14	I2S_SHARED_WS[0] output from I^{2}S pin sharing.	
			0x15	I2S_SHARED_WS[1] output from I^{2}S pin sharing.	
			0x16	ARM_TXEV interrupt event from the CPU.	
			0x17	DEBUG_HALTED from the CPU.	
			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 3

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 390. 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	Reserved.	
		0x12	Reserved.	
		0x13	Reserved.	
		0x14	Reserved.	
		0x15	Reserved.	
		0x16	COMP_OUTPUT output from analog comparator.	
		0x17	I^{2}S_SHARED_WS[0] output from I^{2}S pin sharing.	
		0X18	I^{2}S_SHARED_WS[1] output from I^{2}S 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 391. 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 392. 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 393. 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 394. 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 395. 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).	0xF
		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 396. 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 397. 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 398. 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 399. 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 399. DMA0 request enable register (DMA0_REQ_ENA, offset = 0x740) *…continued*

Bit	Symbol	Description	Reset value
4	REQ_ENA4	Flexcomm Interface 0 RX / I^{2}C Slave.	0x0
5	REQ_ENA5	Flexcomm Interface 0 TX / I^{2}C Master.	0x0
6	REQ_ENA6	Flexcomm Interface 1 RX / I^{2}C Slave.	0x0
7	REQ_ENA7	Flexcomm Interface 1 TX / I^{2}C Master.	0x0
8	REQ_ENA8	Flexcomm Interface 3 RX / I^{2}C Slave.	0x0
9	REQ_ENA9	Flexcomm Interface 3 TX / I^{2}C Master.	0x0
10	REQ_ENA10	Flexcomm Interface 2 RX / I^{2}C Slave.	0x0
11	REQ_ENA11	Flexcomm Interface 2 TX / I^{2}C Master.	0x0
12	REQ_ENA12	Flexcomm Interface 4 RX / I^{2}C Slave.	0x0
13	REQ_ENA13	Flexcomm Interface 4 TX / I^{2}C Master.	0x0
14	REQ_ENA14	Flexcomm Interface 5 RX / I^{2}C Slave.	0x0
15	REQ_ENA15	Flexcomm Interface 5 TX / I^{2}C Master.	0x0
16	REQ_ENA16	Flexcomm Interface 6 RX / I^{2}C Slave.	0x0
17	REQ_ENA17	Flexcomm Interface 6 TX / I^{2}C Master.	0x0
18	REQ_ENA18	Flexcomm Interface 7 RX / I^{2}C Slave.	0x0
19	REQ_ENA19	Flexcomm Interface 7 TX / I^{2}C 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 400. 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 401. 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 402. 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^{2}C Slave.	0x0
5	REQ_ENA5	Flexcomm Interface 0 TX / I^{2}C Master.	0x0
6	REQ_ENA6	Flexcomm Interface 1 RX /I^{2}C Slave.	0x0
7	REQ_ENA7	Flexcomm Interface 1 TX / I^{2}C Master.	0x0
8	REQ_ENA8	Flexcomm Interface 3 RX / I^{2}C Slave.	0x0
9	REQ_ENA9	Flexcomm Interface 3 TX / I^{2}C 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 403. 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 404. 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 405. 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 406. 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 407. 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 408. 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 409. 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 410. 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

UM11424

Chapter 19: LPC55S0x/LPC550x Pin Interrupt and Pattern Match (PINT)

Rev. 1.5 — 21 December 2023 **User manual**

19.1 How to read this chapter

The pin interrupt generator and the pattern match engine are available on all LPC55S0x/LPC550x devices. The 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 the CPU.
  - 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 388. 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.
  - GPIO pins from port 0 and 1 can be masked in input prior to input multiplexer selection for security purposes. See: Section 43.3.4 “Interrupt, DMA and GPIO: Secure instance and masking”.

- 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 388. 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 LPC55S0x/LPC550x 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 be 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 388. 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.

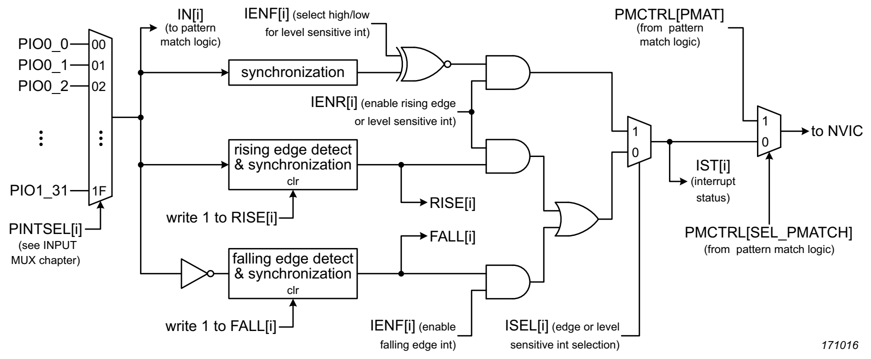

PIO0_0
PIO0_1
PIO0_2
PIO1_31
00
01
02
1F
PINTSEL[i] (see INPUT MUX chapter)
IN[i] (to pattern match logic)
IENF[i] (select high/low for level sensitive int)
synchronization
IENR[i] (enable rising edge or level sensitive int)
rising edge detect & synchronization
clr
write 1 to RISE[i]
RISE[i]
FALL[i]
falling edge detect & synchronization
clr
write 1 to FALL[i]
IENF[i] (enable falling edge int)
ISEL[i] (edge or level sensitive int selection)
PMCTRL[PMAT] (from pattern match logic)
1
0
to NVIC
IST[i] (interrupt status)
PMCTRL[SEL_PMATCH] (from pattern match logic)
171016

Fig 52. 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 by being detected 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 54 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 53. 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.

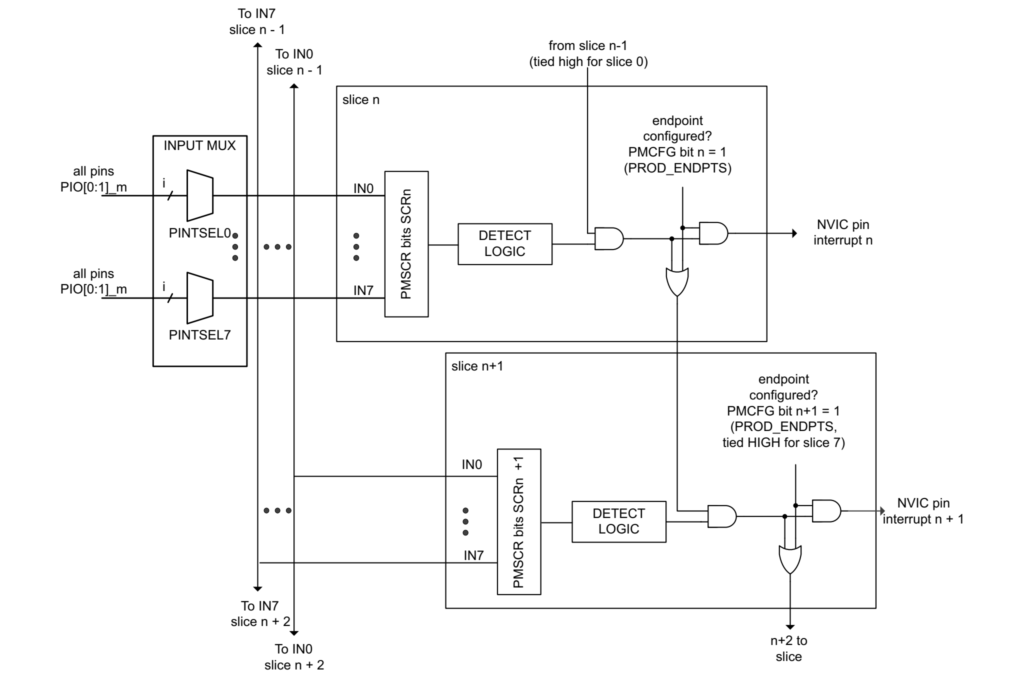


Fig 53. Pattern match engine connections.

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 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 CPU 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 54 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.

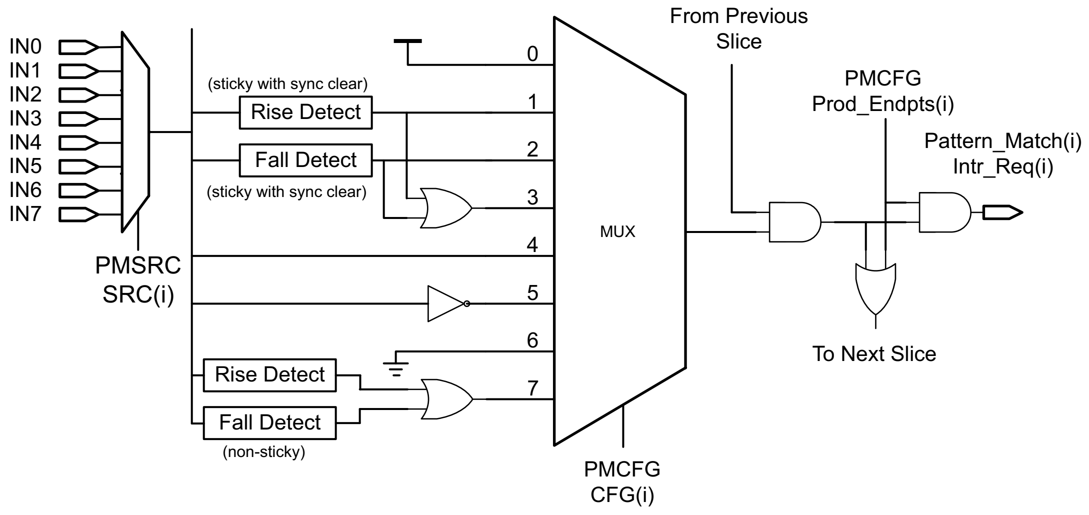


Fig 54. Pattern match bit slice with detect logic

19.5.2.1 Example

The following expression is specified through the registers PMSRC, see Table 423 and PMCFG Table 424:

```
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 CPU. 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 411. 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 412. 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 413. 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 414. 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 415. 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 416. 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 417. 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 418. 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 419. 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 420. 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 421. 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 CPU. This register also allows the current state of any pattern matches to be read.

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

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

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 422. 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 CPU when the specified boolean expression evaluates to true.	0
		0	Disabled. RXEV output to the CPU is disabled.	
		1	Enabled. RXEV output to the CPU is enabled.	
23:2	-	-	Reserved. Do not write 1s to unused bits.	0
31:24	PMAT	-	This field displays the current state of pattern matches. A 1 in any bit of this field indicates that the corresponding product term is matched by the current state of the appropriate inputs.	0x0

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 423. 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 423. 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 423. 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 424. 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 424. 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 424. 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 424. 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 424. 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 425.

Table 425. 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 423.
  - 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 424.
  - 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 422.
  - 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

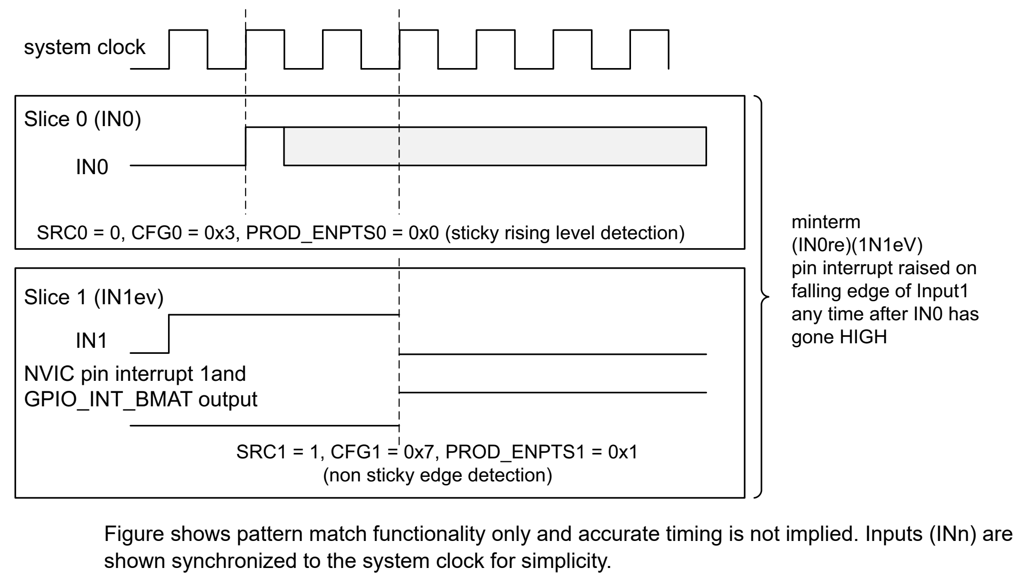

system clock
Slice 0 (IN0)
IN0
SRC0 = 0, CFG0 = 0x3, PROD_ENPTS0 = 0x0 (sticky rising level detection)
Slice 1 (IN1ev)
IN1
NVIC pin interrupt 1and GPIO_INT_BMAT output
SRC1 = 1, CFG1 = 0x7, PROD_ENPTS1 = 0x1 (non sticky edge detection)
minterm (IN0re)(1N1eV) pin interrupt raised on falling edge of Input1 any time after IN0 has gone HIGH
Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 55. Pattern match engine examples: sticky edge detect

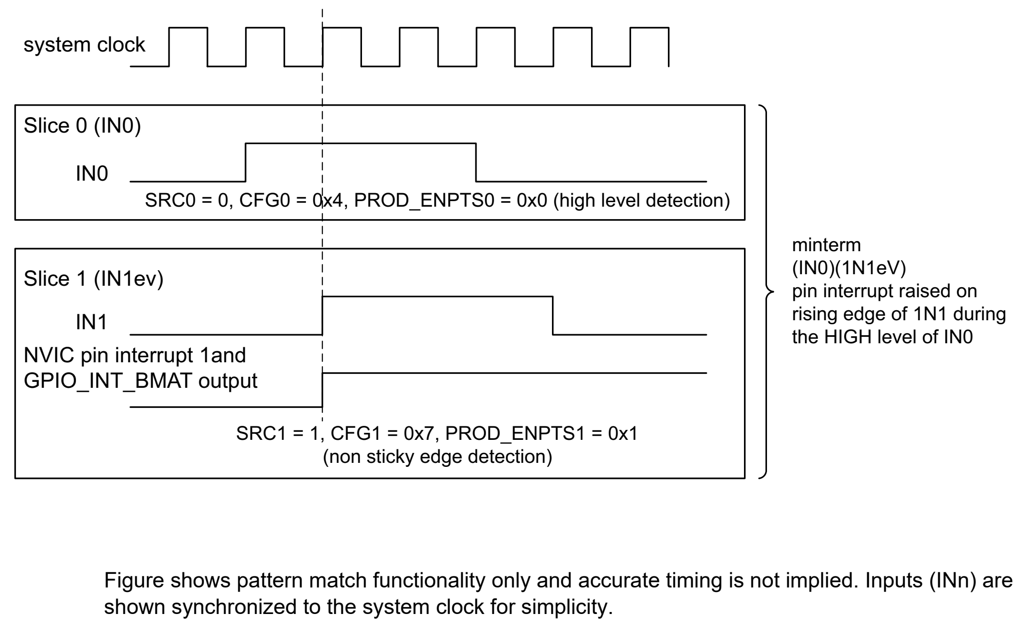

system clock
Slice 0 (IN0)
IN0
SRC0 = 0, CFG0 = 0x4, PROD_ENPTS0 = 0x0 (high level detection)
Slice 1 (IN1ev)
IN1
NVIC pin interrupt 1and GPIO_INT_BMAT output
SRC1 = 1, CFG1 = 0x7, PROD_ENPTS1 = 0x1 (non sticky edge detection)
minterm (IN0)(1N1eV) pin interrupt raised on rising edge of 1N1 during the HIGH level of IN0
Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 56. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true

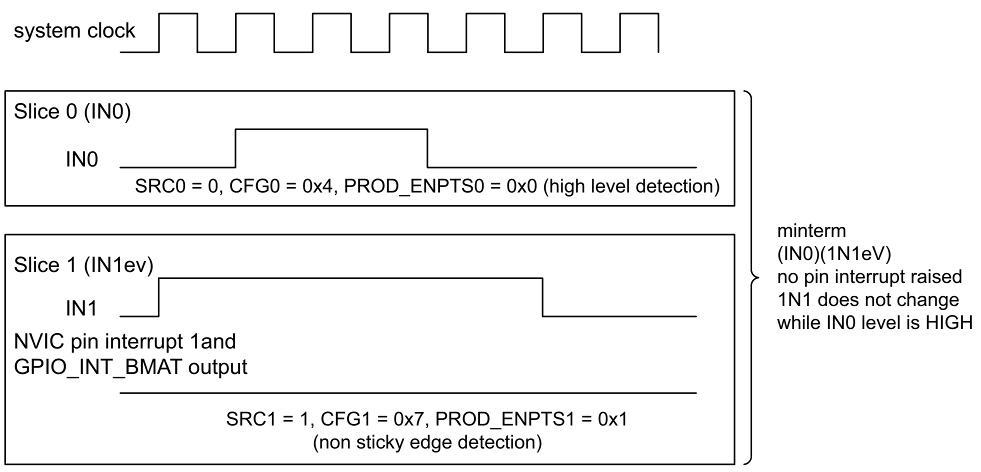

system clock
Slice 0 (IN0)
IN0
SRC0 = 0, CFG0 = 0x4, PROD_ENPTS0 = 0x0 (high level detection)
Slice 1 (IN1ev)
IN1
NVIC pin interrupt 1and GPIO_INT_BMAT output
SRC1 = 1, CFG1 = 0x7, PROD_ENPTS1 = 0x1 (non sticky edge detection)
minterm (IN0)(1N1eV) no pin interrupt raised 1N1 does not change while IN0 level is HIGH

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

Fig 57. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false

UM11424

Chapter 20: LPC55S0x/LPC550x Secure pin interrupt and pattern match (Secure PINT)

Rev. 1.5 — 21 December 2023 **User manual**

20.1 How to read this chapter

The pin interrupt generator and the pattern match engine are available on all LPC55S0x/LPC550x 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 CPU.
  - 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 388). 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 388. 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 LPC55S0x/LPC550x 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 388. 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.

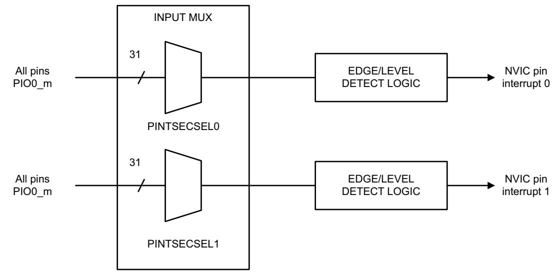


Fig 58. Secure pin interrupt connections.

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 60 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 59. 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.

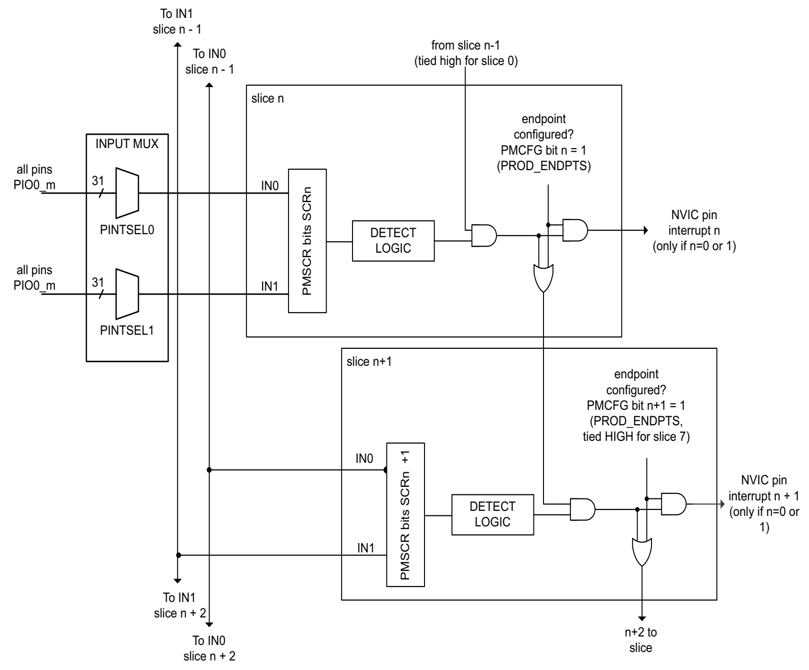


Fig 59. Pattern match engine connections

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 core 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 CPU 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 60.

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.

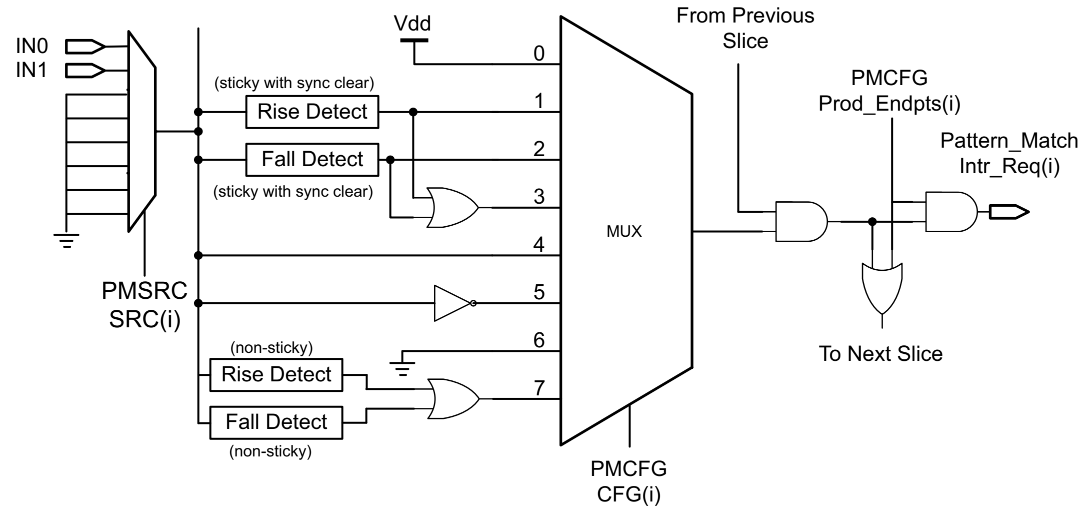


Fig 60. Secure pattern match bit slice with detect logic

20.5.2.1 Example

Assume the expression: (IN0)v~(IN1) + (IN0)^(IN1) is specified through the registers PMSRC Table 438 and PMCFG Table 439. 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 426. 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 427. 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 428. 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 429. 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 430. 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 431. 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 432. 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 433. 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 434. 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 435. 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 436. 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 CPU. This register also allows the current state of any pattern matches to be read.

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

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

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 437. 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 CPU when the specified boolean expression evaluates to true.	0
		0	Disabled. RXEV output to the CPU are disabled.	
		1	Enabled. RXEV output to the CPU 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 438. 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 438. 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 contributes 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 439. 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 439. 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 439. 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 439. 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 439. 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 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 440.

Table 440. 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 438:
  - 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 439.
  - 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 437.
  - 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 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, 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

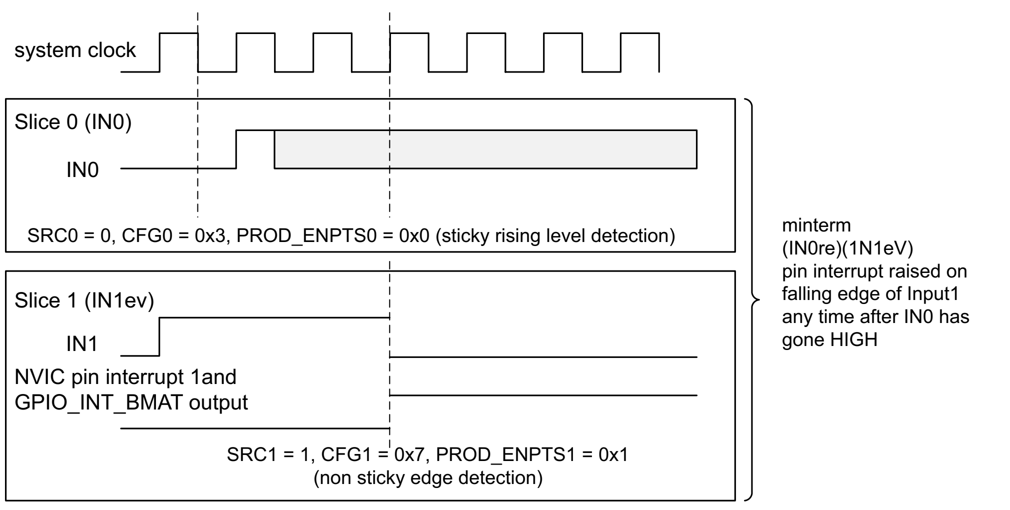

system clock
Slice 0 (IN0)
IN0
SRC0 = 0, CFG0 = 0x3, PROD_ENPTS0 = 0x0 (sticky rising level detection)
Slice 1 (IN1ev)
IN1
NVIC pin interrupt 1and GPIO_INT_BMAT output
SRC1 = 1, CFG1 = 0x7, PROD_ENPTS1 = 0x1 (non sticky edge detection)
minterm (IN0re)(1N1eV) pin interrupt raised on falling edge of Input1 any time after IN0 has gone HIGH

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: sticky edge detect

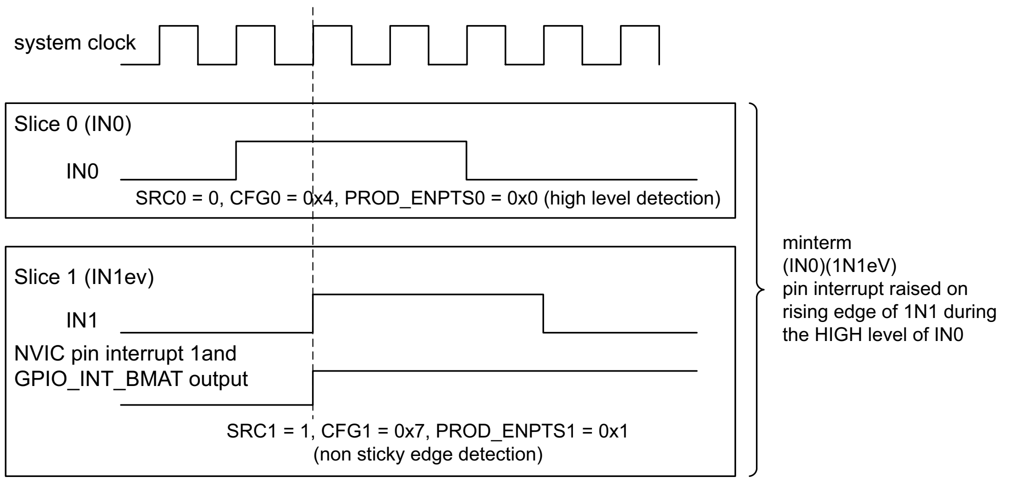

system clock
Slice 0 (IN0)
IN0
SRC0 = 0, CFG0 = 0x4, PROD_ENPTS0 = 0x0 (high level detection)
Slice 1 (IN1ev)
IN1
NVIC pin interrupt 1and GPIO_INT_BMAT output
SRC1 = 1, CFG1 = 0x7, PROD_ENPTS1 = 0x1 (non sticky edge detection)
minterm (IN0)(1N1eV) pin interrupt raised on rising edge of 1N1 during the HIGH level of IN0

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 true

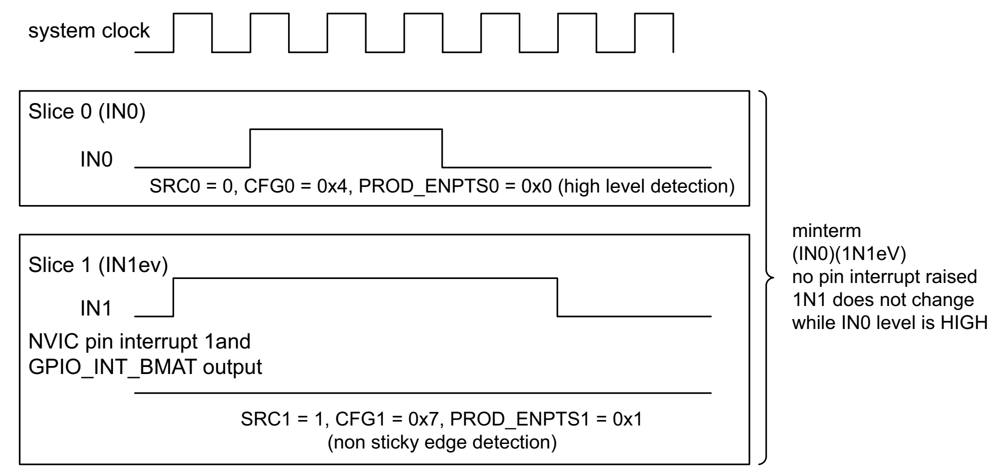


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 63. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false

UM11424

Chapter 21: LPC55S0x/LPC550x:Group GPIO Input Interrupt (GINT0/1)

Rev. 1.5 — 21 December 2023 **User manual**

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 CPU 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 441. 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 442. 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 443. 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 444. 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.

Chapter 22: LPC55S0x/LPC550x DMA controller

22.1 How to read this chapter

The DMA controllers are available on all LPC55S0x/LPC550x 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: 10 channels with multiplexers for 15 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 "LPC55S0x/LPC550x 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, a DMA operation must be triggered before any transfer occurs. This triggering can be performed 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".

The SDMA block interfaces with peripherals that support DMA requests and with 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 (except possibly for those that do not support DMA or contain their own DMA engine), and all APB peripherals.

Two DMAs are available for TZ enabled devices that support DMA operations for secure, and non-secure threads. One DMA is available for non-secure mode while the other can be programmed by ROMCode to serve as the secure DMA. The number of available channels and trigger MUX available on the second DMA is reduced. Also, to protect Secure DMA from insecure requests or triggers, a special masking mechanism is implemented that enables DMA access for a particular device only if the mask is securely disabled.

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

The following figure (Figure 64) shows a block diagram of the DMA architecture.

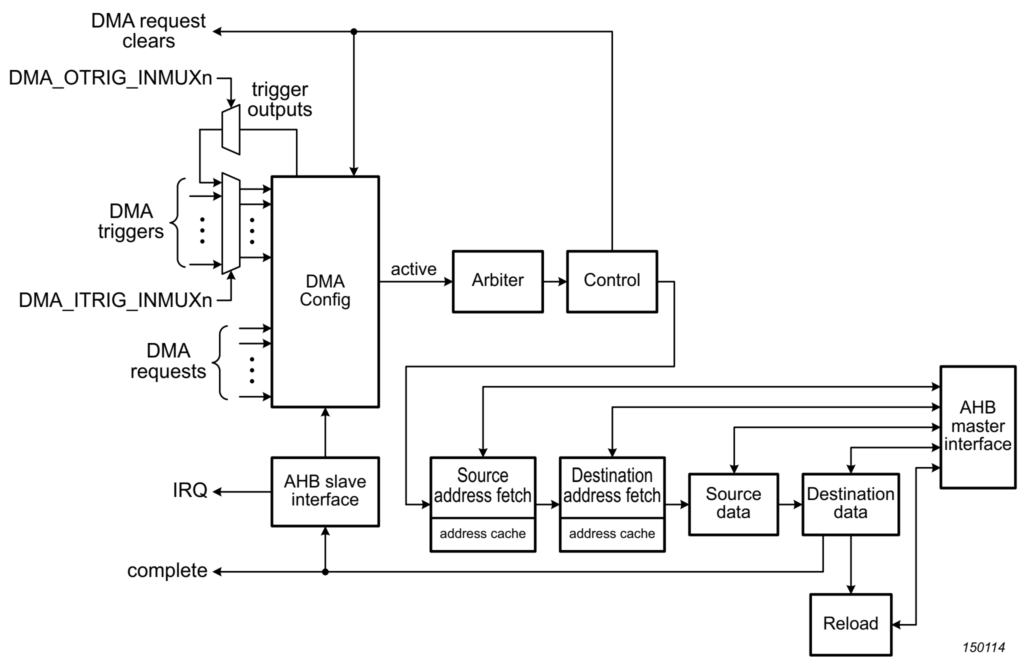


Fig 64. DMA block diagram

22.5.1 DMA requests and triggers

In general, DMA requests are intended to pace transfers to match what the peripheral (including any available FIFO) is capable of processing. 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 445.

Triggers start the transfer. In typical cases, only a software trigger is 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 takes place for any particular DMA channel unless that channel has been triggered, either by software or hardware. DMA triggers are summarized in Table 445.

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 "LPC55S0x/LPC550x 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 request 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 445 and Table 446.

Table 445. 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^{2}C monitor function.

Table 446. 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 I^{2}C monitor function.

22.5.1.1.1 DMA with I^{2}C monitor mode

The I^{2}C monitor function may be used with DMA if one of the channels related to the same Flexcomm Interface is available.

Table 447. DMA with the I^{2}C

I^{2}C Master DMA	I^{2}C Slave DMA	I^{2}C monitor DMA
Not enabled	-	If I^{2}C Monitor DMA is enabled, it will use the DMA channel for the Master function other same Flexcomm Interface.
Enabled	Not enabled	If I^{2}C Monitor is DMA enabled, it will use the DMA channel for the Slave function of the same Flexcomm Interface.
Enabled	Enabled	The I^{2}C 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 448 and Chapter 12 "LPC55xx Input multiplexing (INPUT MUX)".

Table 448. 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 448. 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 445, 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 449.

Table 449. 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 a 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 than the channel descriptor, as shown in Table 450. 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 450. 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

Using a single buffer is usually reserved for memory to memory moves, and peripheral DMAs that occurs only occasionally and must be set up for each transfer. For this kind of operation, only the initial channel descriptor shown in Table 451 is needed.

Table 451. 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 452 shows an example of descriptors for ping-pong from a peripheral to two buffers in memory.

Table 452. 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.

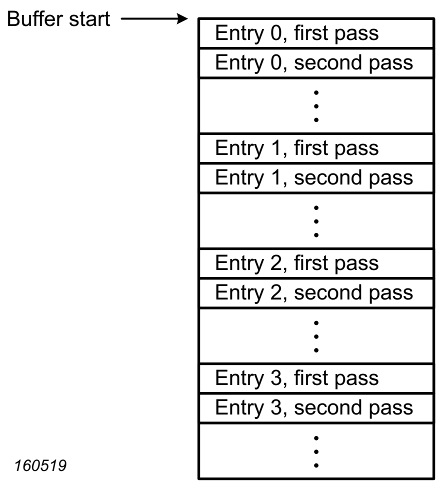


Fig 65. Interleaved transfer in a single buffer

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 452. 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, set up channels x and y for channel chaining as described above. In addition, the following points should be considered.

- 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 modes

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 453. 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

Table 453. 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
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
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

Table 453. 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
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
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

Table 453. 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
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
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					
CFG20	R/W	0x550	Configuration register for DMA channel 21.	0	22.6.16
CTLSTAT20	RO	0x554	Control and status register for DMA channel 21.	0	22.6.17
XFERCFG20	R/W	0x558	Transfer configuration register for DMA channel 21.	0	22.6.18
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 454. 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 455. 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
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 456. 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 457. Only the descriptors for channels defined at extraction are used. The contents of each channel descriptor are described in Table 449.

Table 457. 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.

Table 457. Channel descriptor map [1] *…continued*

Offset	Description
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.
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 458. 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 459. Enable clear register 0 (COMMON_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 460. 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 461. 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 462. 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 463. 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 464. 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 465. 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 466. 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 467. 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 468. 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 469. 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 Channel configuration register contains various configuration options for DMA channel n. See Table 471 for a summary of trigger options.

Table 470. 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 470. 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 470. 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 471. 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 472. 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 473. 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 473. 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 473. 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.	-

UM11424

Chapter 23: LPC55S0x/LPC550x SCTimer/PWM (SCT)

Rev. 1.5 — 21 December 2023 User manual

23.1 How to read this chapter

The SCTimer/PWM is available on all LPC55S0x/LPC550x devices.

Remark: For a detailed description of SCTimer/PWM applications and code examples, see Ref. 2 AN11538.

23.2 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 eight single-edge or four 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.

23.3 Basic configuration

Configure the SCT as follows:

- Enable the clock to the SCTimer/PWM (SCT) in the AHBCLKCTRL1 register, see Section 4.5.17 “AHB clock control 1” to enable the register interface and the peripheral clock.
- Clear the SCT peripheral reset using the PRESETCTRL register, see Section 4.5.7 “Peripheral reset control 1”.
- The SCT provides an interrupt to the NVIC, see Chapter 3 “LPC55S0x/LPC550x Nested Vectored Interrupt Controller (NVIC)”.
- SCT inputs are selected from the SCT input multiplexer registers. See Chapter 18 “LPC55S0x/LPC550x 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”.

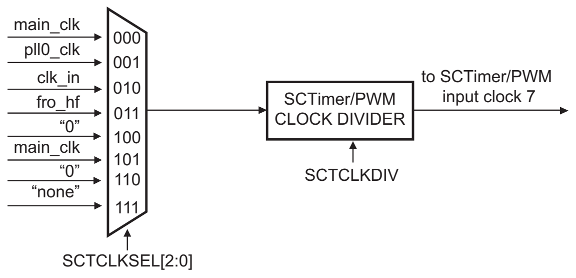


Fig 66. SCTimer/PWM clocking

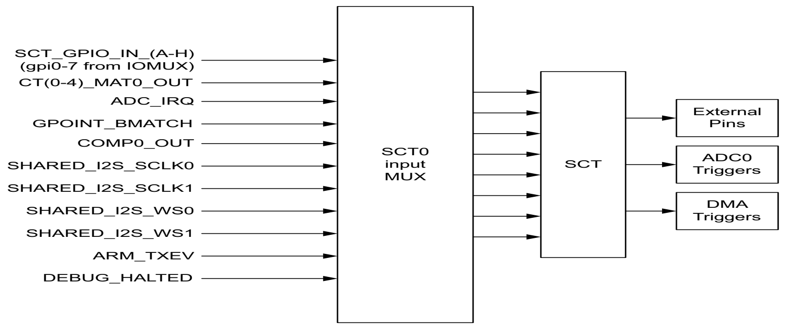


Fig 67. SCTimer/PWM connections

23.4 Pin description

Remark: Availability of inputs or outputs related to a particular peripheral function might be package dependent.

See Chapter 15 “LPC55S0x/LPC550x 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 “LPC55S0x/LPC550x 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 “LPC55S0x/LPC550x Input Multiplexing (INPUTMUX)”.

Table 474. 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, COMP0_OUT, SHARED_I2S_SCLK0, SHARED_I2S_SCLK1, SHARED_I2S_WS0, SHARED_I2S_WS1, ARM_TXEV, DEBUG_HALTED	Figure 67

Table 475. 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		
	SCT0_GPI4	PIO0_4, PIO1_0		
	SCT0_GPI5	PIO0_5, PIO1_1, PIO1_22		
	SCT0_GPI6	PIO0_6, PIO1_2, PIO1_29		
	SCT0_GPI7	PIO0_12		
Internal	-	ADC0 trigger	SCT0 output 4, SCT0 output 5, SCT0 output 6	Table 737
Internal	-	SDMA trigger	SCT_DMA0, SCT_DMA1	Table 445

Table 476. SCT0 pin description (outputs)

Type	Function	Connect to	Use register	Reference
External to pin	SCT0_OUT0	PIO0_2, PIO1_4, PIO1_23	IOCON register for the related pin	Chapter 15
	SCT0_OUT1	PIO0_3, PIO0_18		
	SCT0_OUT2	PIO0_10, PIO0_15, PIO0_19, PIO1_9, PIO1_25		
	SCT0_OUT3	PIO0_22, PIO0_31, PIO1_10		
	SCT0_OUT4	PIO0_23, PIO1_3		
	SCT0_OUT5	PIO0_26		
	SCT0_OUT6	PIO0_27		
	SCT0_OUT7	PIO0_28		
	SCT0_OUT8	PIO0_29		
	SCT0_OUT9	PIO0_30		
Internal	-	ADC0 trigger	SCT0 output 4, SCT0 output 5, SCT0 output 6	Table 737
Internal	-	SDMA trigger	SCT_DMA0, SCT_DMA1	Table 445

Table 477. 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 477. 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 477 and Table 478.

Table 478. 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 recommend ed for SCT outputs.

23.5 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. 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.Software access to 16-bit registers when the LPC55S0x/LPC550x SCT is configured as two 16-bit counters has some limitations, see Section 23.5.1 “Important notes on using the SCT as two 16-bit counters”.

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 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.

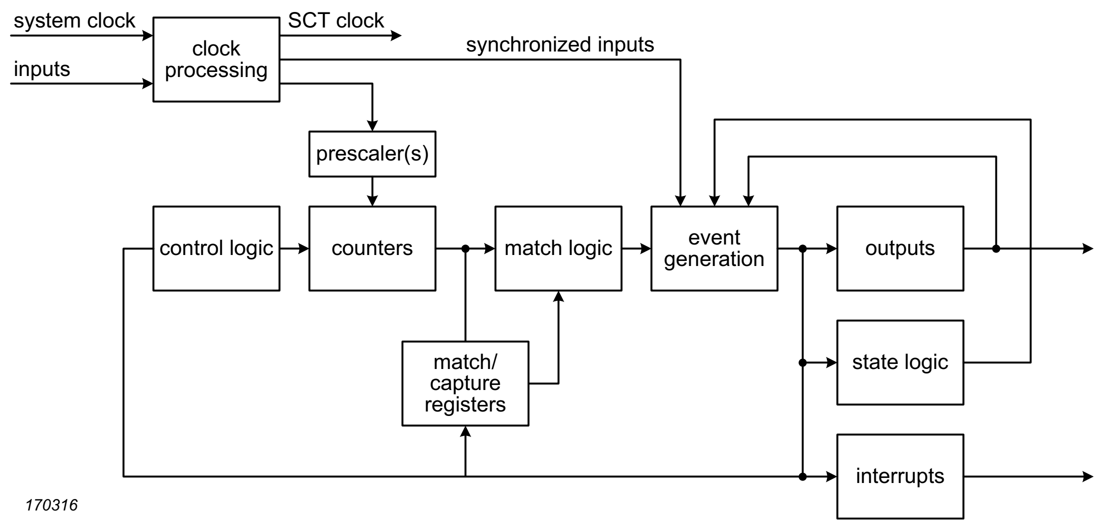


Fig 68. SCTimer/PWM block diagram

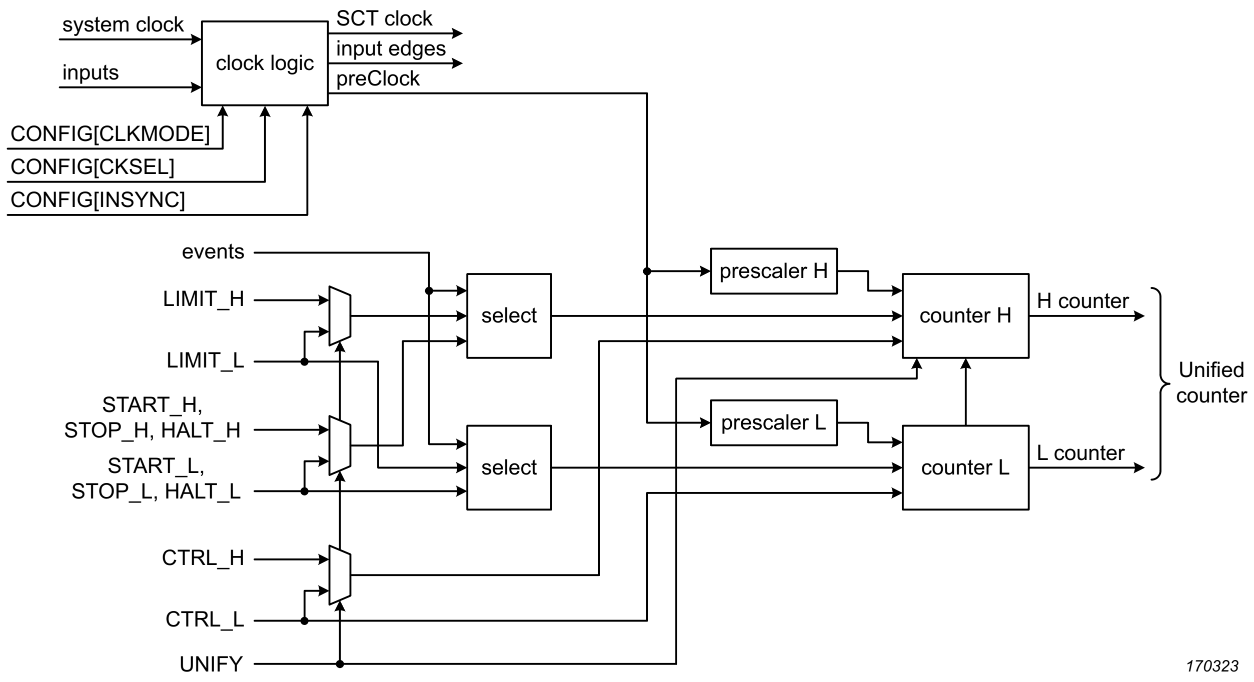


Fig 69. SCTimer/PWM counter and select logic

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

23.5.1 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.

23.6 Register description

The register addresses of the SCTimer/PWM are shown in Table 479. 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 479. Register overview: SCTimer/PWM (base address = 0x4008 5000)

Name	Access	Offset	Description	Reset value	Section
CONFIG	R/W	0x000	SCT configuration register.	0x0001 FE00	23.6.2
CTRL	R/W	0x004	SCT control register.	0x0004 0004	23.6.3
CTRL_L	R/W	0x004	SCT control register low counter 16-bit.	0x0000 0004	23.6.3
CTRL_H	R/W	0x006	SCT control register high counter 16-bit.	0x0000 0004	23.6.3
LIMIT	R/W	0x008	SCT limit event select register.	0x0000 0000	23.6.4
LIMIT_L	R/W	0x008	SCT limit event select register low counter 16-bit.	0x0000 0000	23.6.4
LIMIT_H	R/W	0x00A	SCT limit event select register high counter 16-bit.	0x0000 0000	23.6.4
HALT	R/W	0x00C	SCT halt event select register.	0x0000 0000	23.6.5
HALT_L	R/W	0x00C	SCT halt event select register low counter 16-bit.	0x0000 0000	23.6.5
HALT_H	R/W	0x00E	SCT halt event select register high counter 16-bit.	0x0000 0000	23.6.5
STOP	R/W	0x010	SCT stop event select register.	0x0000 0000	23.6.6
STOP_L	R/W	0x010	SCT stop event select register low counter 16-bit.	0x0000 0000	23.6.6
STOP_H	R/W	0x012	SCT stop event select register high counter 16-bit.	0x0000 0000	23.6.6
START	R/W	0x014	SCT start event select register.	0x0000 0000	23.6.7
START_L	R/W	0x014	SCT start event select register low counter 16-bit.	0x0000 0000	23.6.7
START_H	R/W	0x016	SCT start event select register high counter 16-bit.	0x0000 0000	23.6.7
COUNT	R/W	0x040	SCT counter register.	0x0000 0000	23.6.8
COUNT_L	R/W	0x040	SCT counter register low counter 16-bit.	0x0000 0000	23.6.8

Table 479. 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	23.6.8
STATE	R/W	0x044	SCT state register.	0x0000 0000	23.6.9
STATE_L	R/W	0x044	SCT state register low counter 16-bit.	0x0000 0000	23.6.9
STATE_H	R/W	0x046	SCT state register high counter 16-bit.	0x0000 0000	23.6.9
INPUT	RO	0x048	SCT input register.	0x0000 0000	23.6.10
REGMODE	R/W	0x04C	SCT match/capture mode register.	0x0000 0000	23.6.11
REGMODE_L	R/W	0x04C	SCT match/capture mode register low counter 16-bit.	0x0000 0000	23.6.11
REGMODE_H	R/W	0x04E	SCT match/capture registers mode register high counter 16-bit.	0x0000 0000	23.6.11
OUTPUT	R/W	0x050	SCT output register.	0x0000 0000	23.6.12
OUTPUTDIRCTRL	R/W	0x054	SCT output counter direction control register.	0x0000 0000	23.6.13
RES	R/W	0x058	SCT conflict resolution register.	0x0000 0000	23.6.14
DMAREQ0	R/W	0x05C	SCT DMA request 0 register.	0x0000 0000	23.6.15
DMAREQ1	R/W	0x060	SCT DMA request 1 register.	0x0000 0000	23.6.15
EVEN	R/W	0x0F0	SCT event interrupt enable register.	0x0000 0000	23.6.16
EVFLAG	R/W	0x0F4	SCT event flag register.	0x0000 0000	23.6.17
CONEN	R/W	0x0F8	SCT conflict interrupt enable register.	0x0000 0000	23.6.18
CONFLAG	R/W	0x0FC	SCT conflict flag register.	0x0000 0000	23.6.19
MATCH0 to MATCH15	R/W	0x100 to 0x13C	SCT match value register of match channels 0 to 15; REGMODE0 to REGMODE15 = 0.	0x0000 0000	23.6.20
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	23.6.20
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	23.6.20
CAP0 to CAP15	R/W	0x100 to 0x13C	SCT capture register of capture channel 0 to 15; REGMODE0 to REGMODE15 = 1.	0x0000 0000	23.6.21
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	23.6.21
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	23.6.21
MATCHREL0 to MATCHREL15	R/W	0x200 to 0x23C	SCT match reload value register 0 to 15; REGMODE0 = 0 to REGMODE15 = 0.	0x0000 0000	23.6.22
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	23.6.22
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	23.6.22
CAPCTRL0 to CAPCTRL15	R/W	0x200 to 0x23C	SCT capture control register 0 to 15; REGMODE0 = 1 to REGMODE15 = 1.	0x0000 0000	23.6.23
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	23.6.23
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	23.6.23
EV0_STATE	R/W	0x300	SCT event state register 0.	0x0000 0000	23.6.24
EV0_CTRL	R/W	0x304	SCT event control register 0.	0x0000 0000	23.6.25

Table 479. Register overview: SCTimer/PWM (base address = 0x4008 5000) *…continued*

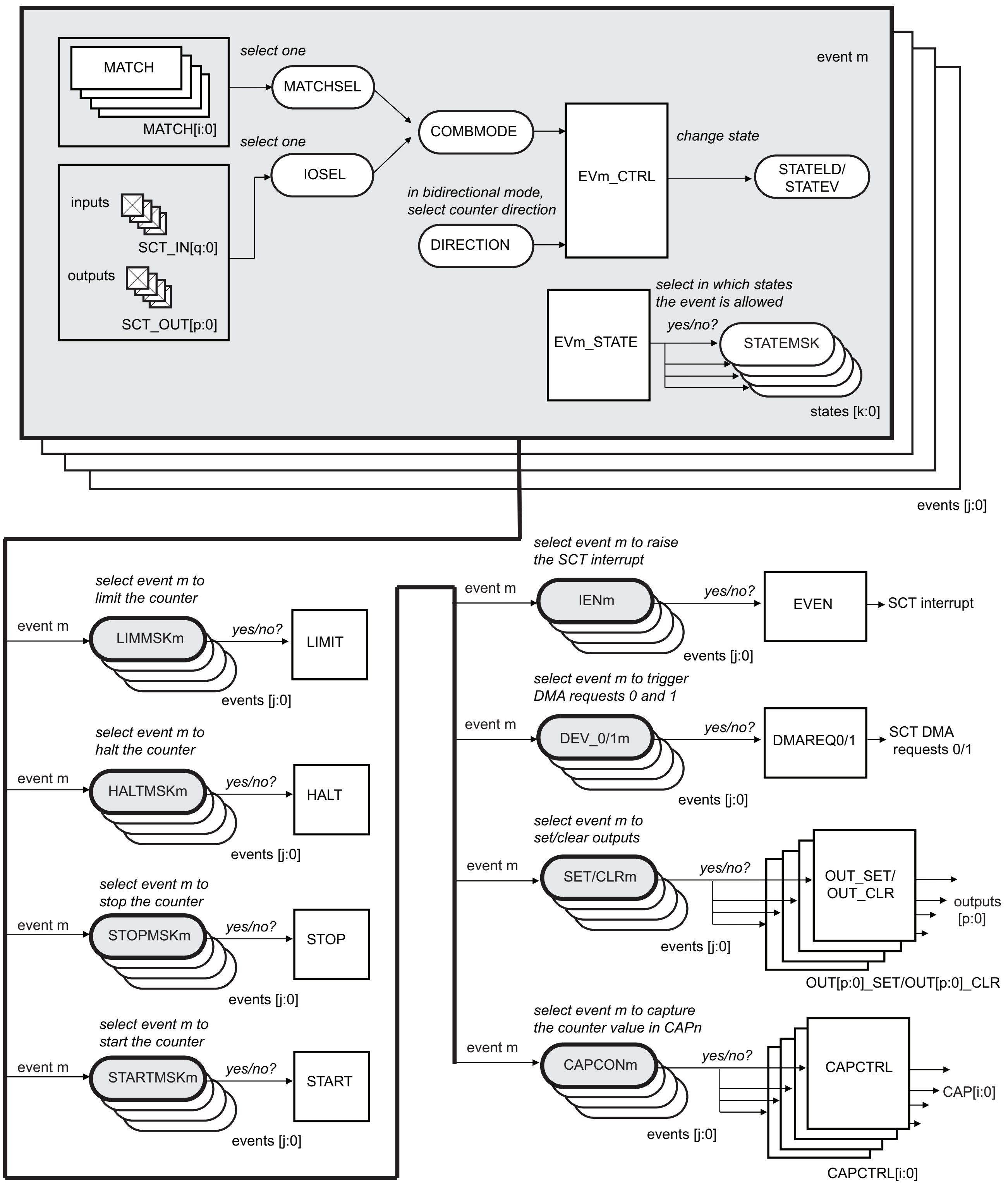
Name	Access	Offset	Description	Reset value	Section
EV1_STATE	R/W	0x308	SCT event state register 1.	0x0000 0000	23.6.24
EV1_CTRL	R/W	0x30C	SCT event control register 1.	0x0000 0000	23.6.25
EV2_STATE	R/W	0x310	SCT event state register 2.	0x0000 0000	23.6.24
EV2_CTRL	R/W	0x314	SCT event control register 2.	0x0000 0000	23.6.25
EV3_STATE	R/W	0x318	SCT event state register 3.	0x0000 0000	23.6.24
EV3_CTRL	R/W	0x31C	SCT event control register 3.	0x0000 0000	23.6.25
EV4_STATE	R/W	0x320	SCT event state register 4.	0x0000 0000	23.6.24
EV4_CTRL	R/W	0x324	SCT event control register 4.	0x0000 0000	23.6.25
EV5_STATE	R/W	0x328	SCT event state register 5.	0x0000 0000	23.6.24
EV5_CTRL	R/W	0x32C	SCT event control register 5.	0x0000 0000	23.6.25
EV6_STATE	R/W	0x330	SCT event state register 6.	0x0000 0000	23.6.24
EV6_CTRL	R/W	0x334	SCT event control register 6.	0x0000 0000	23.6.25
EV7_STATE	R/W	0x338	SCT event state register 7.	0x0000 0000	23.6.24
EV7_CTRL	R/W	0x33C	SCT event control register 7.	0x0000 0000	23.6.25
EV8_STATE	R/W	0x340	SCT event state register 8.	0x0000 0000	23.6.24
EV8_CTRL	R/W	0x344	SCT event control register 8.	0x0000 0000	23.6.25
EV9_STATE	R/W	0x348	SCT event state register 9.	0x0000 0000	23.6.24
EV9_CTRL	R/W	0x34C	SCT event control register 9.	0x0000 0000	23.6.25
EV10_STATE	R/W	0x350	SCT event state register 10.	0x0000 0000	23.6.24
EV10_CTRL	R/W	0x354	SCT event control register 10.	0x0000 0000	23.6.25
EV11_STATE	R/W	0x358	SCT event state register 11.	0x0000 0000	23.6.24
EV11_CTRL	R/W	0x35C	SCT event control register 11.	0x0000 0000	23.6.25
EV12_STATE	R/W	0x360	SCT event state register 12.	0x0000 0000	23.6.24
EV12_CTRL	R/W	0x364	SCT event control register 12.	0x0000 0000	23.6.25
EV13_STATE	R/W	0x368	SCT event state register 13.	0x0000 0000	23.6.24
EV13_CTRL	R/W	0x36C	SCT event control register 13.	0x0000 0000	23.6.25
EV14_STATE	R/W	0x370	SCT event state register 14.	0x0000 0000	23.6.24
EV14_CTRL	R/W	0x374	SCT event control register 14.	0x0000 0000	23.6.25
EV15_STATE	R/W	0x378	SCT event state register 15.	0x0000 0000	23.6.24
EV15_CTRL	R/W	0x37C	SCT event control register 15.	0x0000 0000	23.6.25
OUT0_SET	R/W	0x500	SCT output 0 set register.	0x0000 0000	23.6.26
OUT0_CLR	R/W	0x504	SCT output 0 clear register.	0x0000 0000	23.6.27
OUT1_SET	R/W	0x508	SCT output 1 set register.	0x0000 0000	23.6.26
OUT1_CLR	R/W	0x50C	SCT output 1 clear register.	0x0000 0000	23.6.27
OUT2_SET	R/W	0x510	SCT output 2 set register.	0x0000 0000	23.6.26
OUT2_CLR	R/W	0x514	SCT output 2 clear register.	0x0000 0000	23.6.27
OUT3_SET	R/W	0x518	SCT output 3 set register.	0x0000 0000	23.6.26
OUT3_CLR	R/W	0x51C	SCT output 3 clear register.	0x0000 0000	23.6.27
OUT4_SET	R/W	0x520	SCT output 4 set register.	0x0000 0000	23.6.26
OUT4_CLR	R/W	0x524	SCT output 4 clear register.	0x0000 0000	23.6.27
OUT5_SET	R/W	0x528	SCT output 5 set register.	0x0000 0000	23.6.26

Table 479. 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	23.6.27
OUT6_SET	R/W	0x530	SCT output 6 set register.	0x0000 0000	23.6.26
OUT6_CLR	R/W	0x534	SCT output 6 clear register.	0x0000 0000	23.6.27
OUT7_SET	R/W	0x538	SCT output 7 set register.	0x0000 0000	23.6.26
OUT7_CLR	R/W	0x53C	SCT output 7 clear register.	0x0000 0000	23.6.27
OUT8_SET	R/W	0x540	SCT output 8 set register.	0x0000 0000	23.6.26
OUT8_CLR	R/W	0x544	SCT output 8 clear register.	0x0000 0000	23.6.27
OUT9_SET	R/W	0x548	SCT output 9 set register.	0x0000 0000	23.6.26
OUT9_CLR	R/W	0x54C	SCT output 9 clear register.	0x0000 0000	23.6.27

23.6.1 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 70 shows the registers and register bits that need to be configured for each event.


MATCH
MATCH[i:0]
select one
MATCHSEL
select one
IOSEL
inputs
SCT_IN[q:0]
outputs
SCT_OUT[p:0]
COMBMODE
in bidirectional mode, select counter direction
DIRECTION
EVm_CTRL
change state
STATELD/ STATEV
event m
select in which states the event is allowed
EVm_STATE
yes/no?
STATEMSK
states [k:0]
events [j:0]
select event m to limit the counter
event m
LIMMSKm
yes/no?
LIMIT
events [j:0]
select event m to halt the counter
HALTMSKm
HALT
select event m to stop the counter
STOPMSKm
STOP
select event m to start the counter
STARTMSKm
START
select event m to raise the SCT interrupt
IENm
EVEN
SCT interrupt
select event m to trigger DMA requests 0 and 1
DEV_0/1m
DMAREQ0/1
SCT DMA requests 0/1
select event m to set/clear outputs
SET/CLRm
OUT_SET/ OUT_CLR
outputs [p:0]
OUT[p:0]_SET/OUT[p:0]_CLR
select event m to capture the counter value in CAPn
CAPCONm
CAPCTRL
CAP[i:0]
CAPCTRL[i:0]

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 70. SCT event configuration and selection registers

23.6.1.1 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.

23.6.1.2 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.

23.6.1.3 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.

23.6.1.4 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.

23.6.1.5 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.

23.6.1.6 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.

23.6.1.7 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.

23.6.1.8 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.

23.6.1.9 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.

23.6.2 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 480. 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 480. 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.	-

23.6.3 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 481. 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.	-

Table 481. SCT control register (CTRL, offset = 0x004) *…continued*

Bit	Symbol	Value	Description	Reset value
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
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.	-

23.6.4 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 480.

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 482. 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

23.6.5 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 481.

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 483. 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

23.6.6 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 484. 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

23.6.7 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 485. 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

23.6.8 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. In this case, the L and H registers count independently under the control of the other registers. The L registers can also be written individually, but the H register must be written as a word along with the L register.

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 486. 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

23.6.9 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. 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 23.6.24 "SCT event enable registers 0 to 15" and Section 23.6.25 "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 487. 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.	-

23.6.10 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 488. 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.	-

23.6.11 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 23.6.20 "SCT match registers 0 to 15 (REGMODEn bit = 0)") or as a capture register, see Section 23.6.21 "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 23.6.22 "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 23.6.23 "SCT capture control registers 0 to 15 (REGMODEn bit = 1)". REGMODE_H is used only when the UNIFY bit is 0.

Table 489. 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

23.6.12 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 490. 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	-

23.6.13 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 23.6.26 "SCT output set registers 0 to 9" and Section 23.6.27 "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 491. 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.

23.6.14 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 492. 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.	
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.

23.6.15 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 493. 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 494. 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

23.6.16 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 23.6.17 "SCT event flag register" is also set.

Table 495. 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.	-

23.6.17 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 496. 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.	-

23.6.18 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 497. 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.	

23.6.19 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 498. 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

23.6.20 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 is 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 23.6.4 “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 499. 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

23.6.21 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 500. 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

23.6.22 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 501. 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

23.6.23 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 written 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. Based on a selected event, the capture registers can be loaded with the current counter value when the event occurs.

Each capture control register (L, H, or unified 32-bit) controls which events cause the load of corresponding capture register from the counter.

Table 502. 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

23.6.24 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 503. 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.	-

23.6.25 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 504. 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
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

Table 504. 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
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.	-

23.6.26 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 505. 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.	-

23.6.27 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 506. 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.	-

23.7 Functional description

23.7.1 Match logic

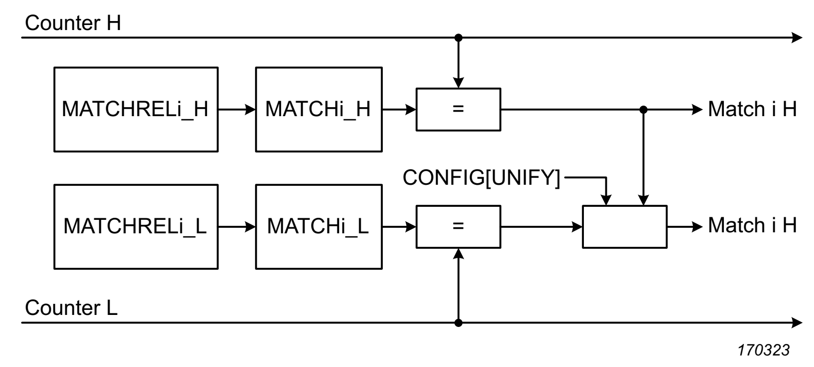


Fig 71. Match logic

23.7.2 Capture logic

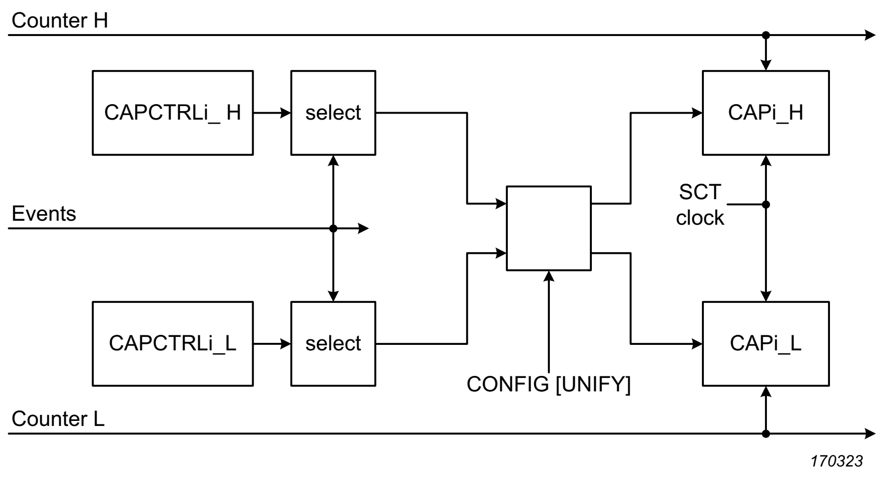


Fig 72. Capture logic

23.7.3 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.

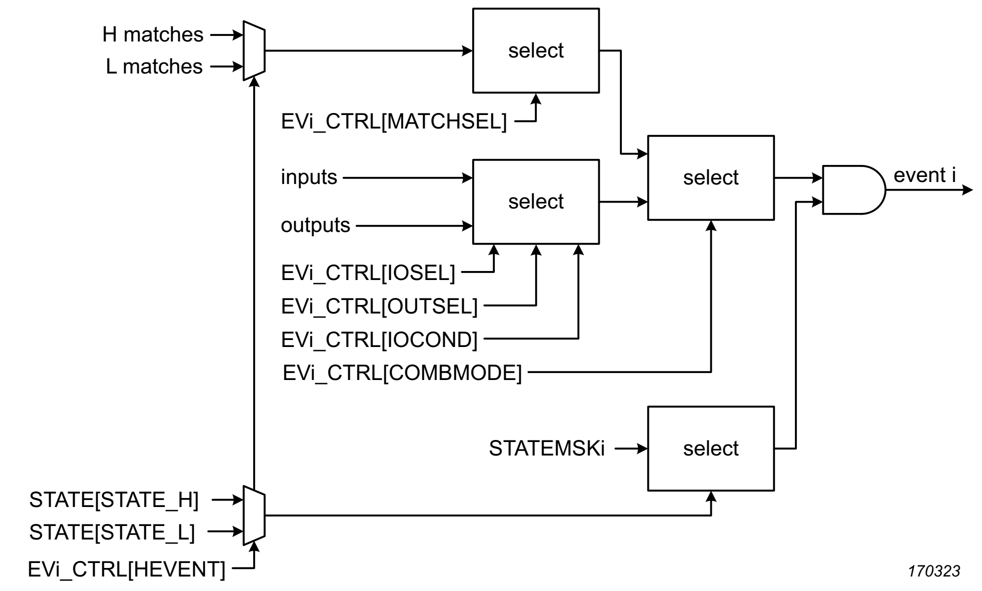


Fig 73. Event selection

23.7.4 Output generation

Figure 74 shows one output slice of the SCT.

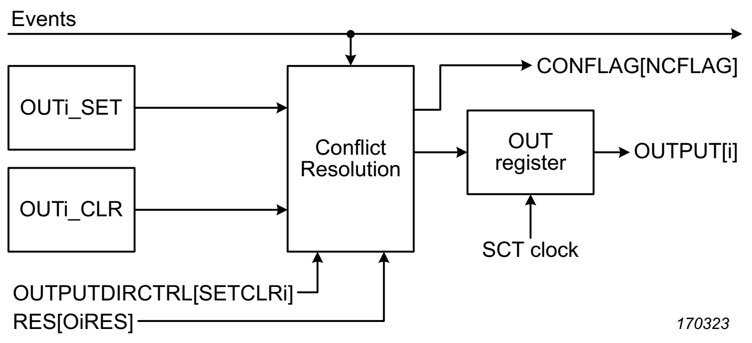


Fig 74. Output slice i

23.7.5 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.

23.7.6 Interrupt generation

The SCT generates one interrupt to the NVIC.

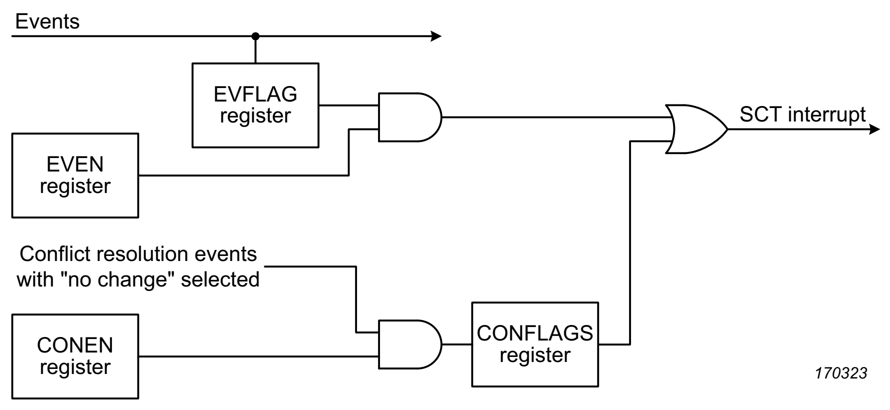


Fig 75. SCT interrupt generation

23.7.7 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 23.7.8 “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.

23.7.8 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 507 shows when the various kinds of events can occur.

Table 507. 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).

23.7.9 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 23.7.10 "Configure the SCT".
- To start, run, and stop the SCT, see Section 23.7.11 "Run the SCT".
- To configure the SCT as simple event controlled counter/timer, see Section 23.7.12 "Configure the SCT without using states".

23.7.10 Configure the SCT

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

23.7.10.1 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.

23.7.10.2 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.

23.7.10.3 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.

23.7.10.4 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.

23.7.10.5 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.

23.7.11 Run the SCT

1. Configure the SCT, see Section 23.7.10 “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.

23.7.12 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.

23.7.13 SCT PWM example

Figure 76 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.

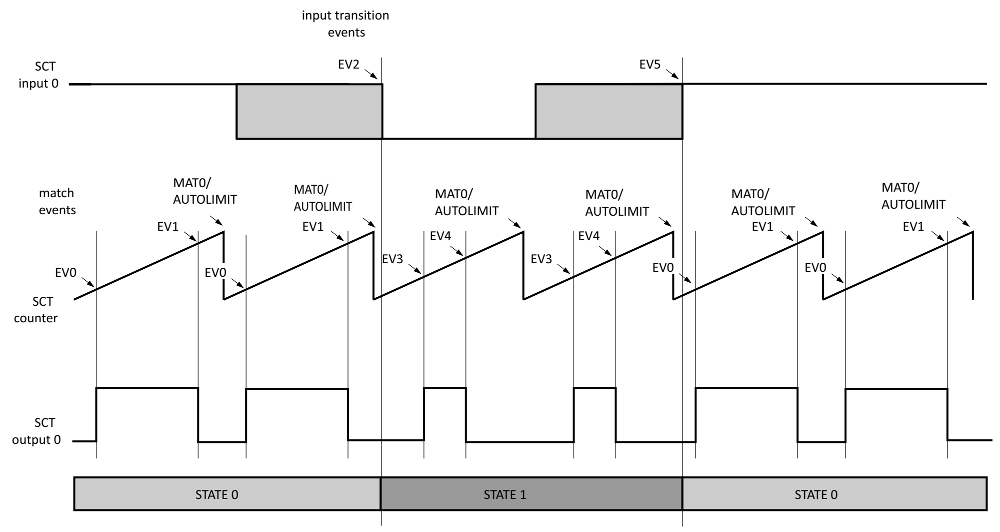


Fig 76. SCT configuration example

This application of the SCT uses the following configuration (all register values not listed in Table 508 are set to their default values):

Table 508. 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 508. 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 508. 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.

UM11424

Chapter 24: LPC55S0x/LPC550x Standard counter/timers (CTIMER0 - 4)

Rev. 1.5 — 21 December 2023 **User manual**

24.1 How to read this chapter

These five standard timers are available on all LPC55S0x/LPC550x devices.

24.2 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 “CTimer 0 clock source select” through Section 4.5.33 “CTimer 4 clock source select register”.
- 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.

24.3 Basic configuration

- Select clock source. See Section 4.5.27 “CTimer 0 clock source select” to Section 4.5.31 “CTimer 4 clock source select register”.
- Set the appropriate bits to enable clocks to timers that will be used AHBCLKCTRL registers, see Section 4.5.17 “AHB clock control 1” and Section 4.5.18 “AHB clock control 2”.
- Clear the timer reset using the PRESETCTRL registers, see Section 4.5.7 “Peripheral reset control 1” and Section 4.5.8 “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 “LPC55S0x/LPC550x I/O Pin Configuration (IOCON)”.
- Interrupts: See register MCR (Table 516) and CCR (Table 518) 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 “LPC55S0x/LPC550x DMA controller”.

24.4 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).

24.4.1 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 24.6.11 “Count control register”.

24.4.2 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.

24.4.3 Applications

- Interval timer for counting internal events.
- Pulse Width Modulator via match outputs.
- Pulse Width Demodulator via capture input.
- Free running timer.

24.4.4 Architecture

The block diagram for the timers is shown in Figure 77.

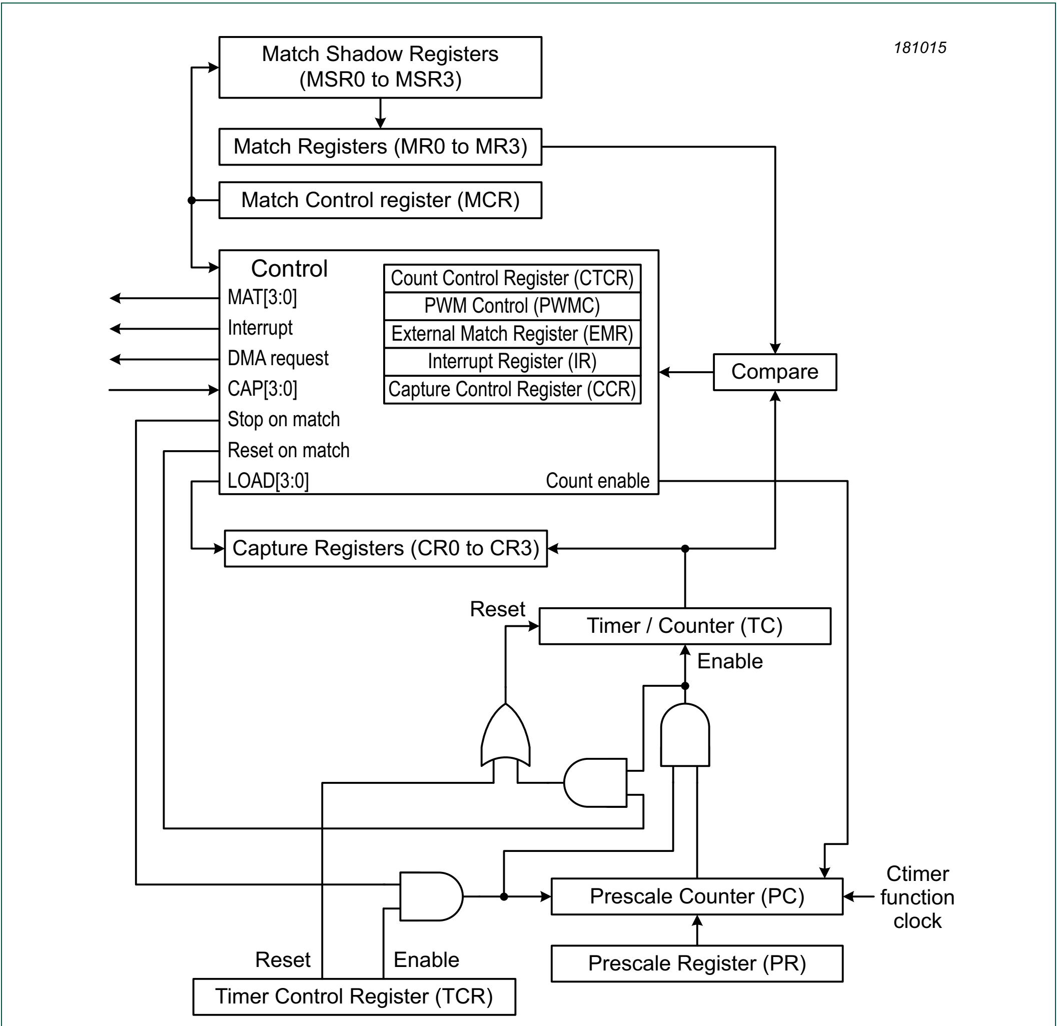


Fig 77. 32-bit counter/timer block diagram

24.4.5 Peripheral input multiplexers for CTimers

See Chapter 15 “LPC55S0x/LPC550x I/O Pin Configuration (IOCON)”.

The pen assignments for the peripheral input multiplexer is shown in figure Figure 78

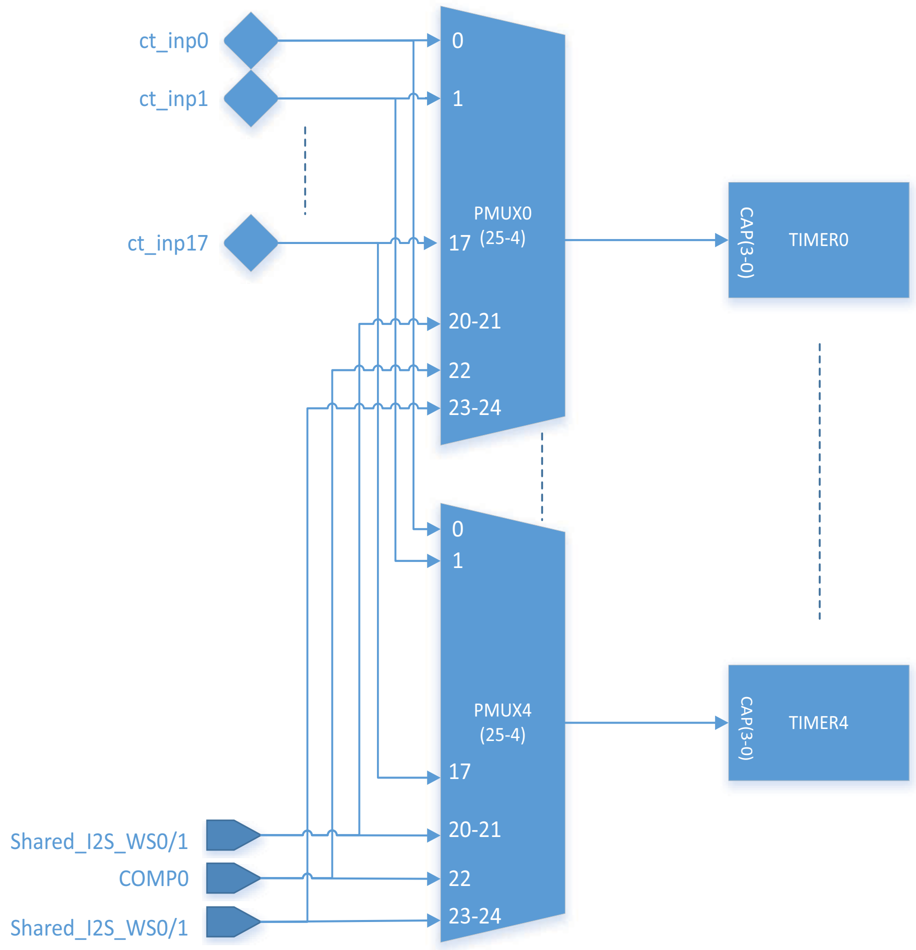


Fig 78. Pen assignments for the Peripheral input multiplexer for CTimers

24.5 Pin description

Table 509 gives a brief summary of each of the Timer/Counter related pins.

Table 509. 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 24.6.11 “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.

24.5.1 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 “LPC55S0x/LPC550x I/O Pin Configuration (IOCON)”. Note that match conditions may be used internally without the use of a device pin.

24.6 Register description

Each Timer/Counter contains the registers shown in Table 510.

Table 510. 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	24.6.1
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	24.6.2
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	24.6.3
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	24.6.4
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	24.6.5
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	24.6.6
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	24.6.7
MR1	R/W	0x1C	Match Register 1. See MR0 description.	0	24.6.7
MR2	R/W	0x20	Match Register 2. See MR0 description.	0	24.6.7
MR3	R/W	0x24	Match Register 3. See MR0 description.	0	24.6.7
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	24.6.8
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	24.6.9
CR1	RO	0x30	Capture Register 1. See CR0 description.	0	24.6.9
CR2	RO	0x34	Capture Register 2. See CR0 description.	0	24.6.9
CR3	RO	0x38	Capture Register 3. See CR0 description.	0	24.6.9
EMR	R/W	0x3C	External Match Register. The EMR controls the match function and the external match pins.	0	24.6.10
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	24.6.11
PWMC	R/W	0x74	PWM Control Register. The PWMC enables PWM mode for the external match pins.	0	24.6.12
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	24.6.13

Table 510. 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	24.6.13
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	24.6.13
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	24.6.13

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

24.6.1 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 511. 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.	-

24.6.2 Timer control register

The Timer Control Register (TCR) is used to control the operation of the Timer/Counter.

Table 512. 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 512. 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

24.6.3 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 513. Timer counter registers (TC, offset 0x08)

Bit	Symbol	Description	Reset value
31:0	TCVAL	Timer counter value.	0

24.6.4 Pre-scale register

The 32-bit Pre-scale register specifies the maximum value for the Pre-scale Counter.

Table 514. Timer pre scale registers (PR, offset 0x00C)

Bit	Symbol	Description	Reset value
31:0	PRVAL	Pre-scale counter value.	0

24.6.5 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 515. Timer pre-scale counter registers (PC, offset 0x010)

Bit	Symbol	Description	Reset value
31:0	PCVAL	Pre-scale counter value.	0

24.6.6 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 516. 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

24.6.7 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 517. Timer match registers (MR[0:3], offset [0x018:0x024])

Bit	Symbol	Description	Reset value
31:0	MATCH	Timer counter match value.	0

24.6.8 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 518. 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

24.6.9 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 519. Timer capture registers (CR[0:3], offsets [0x02C:0x038])

Bit	Symbol	Description	Reset value
31:0	CAP	Timer counter capture value.	0

24.6.10 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 24.7.2 “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 24.7.1 “Rules for single edge controlled PWM outputs”.

Table 520. 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 520. 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

24.6.11 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 521. 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

24.6.12 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 522. 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

24.6.13 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 523. Timer match shadow registers (MSR[0:3], offset [0x78:0x84])

Bit	Symbol	Description	Reset value
31:0	SHADOW	Timer counter match shadow value.	0x0

24.7 Functional description

Figure 79 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 80 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.

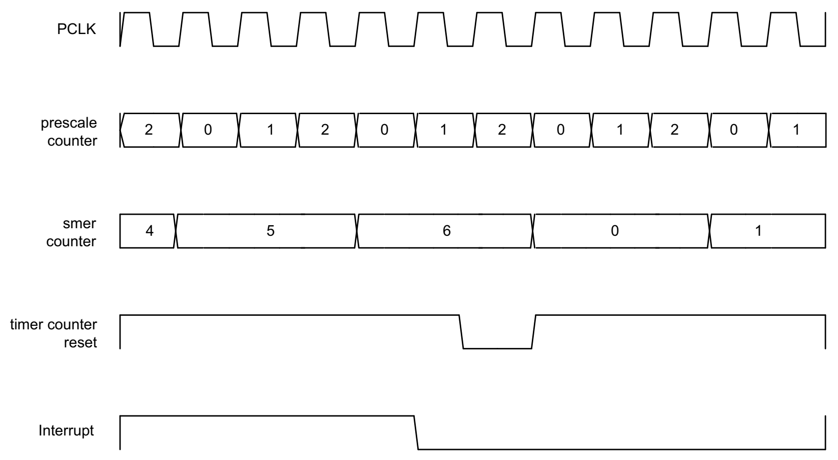


Fig 79. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

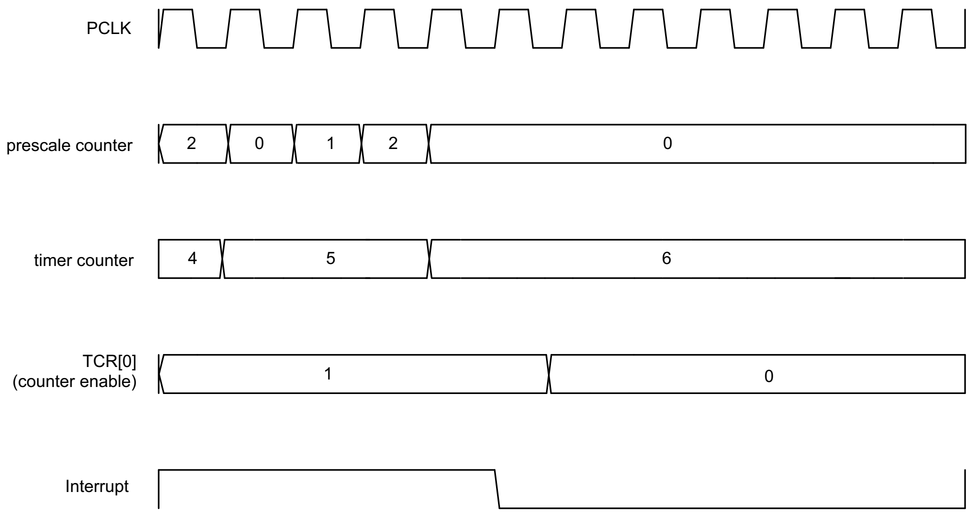


Fig 80. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

24.7.1 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.

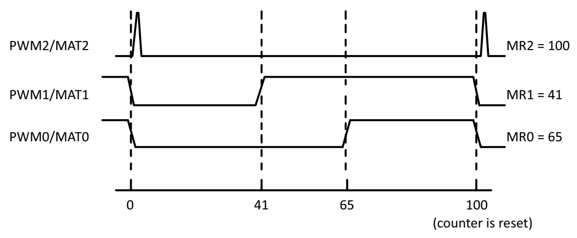


Fig 81. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

24.7.2 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 24.6.1 “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.

UM11424

Chapter 25: LPC55S0x/LPC550x Micro-tick Timer (UTICK)

Rev. 1.5 — 21 December 2023 User manual

25.1 How to read this chapter

The Micro-tick timer is available on all LPC55S0x/LPC550x devices.

25.2 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.

25.3 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 "LPC55S0x/LPC550x 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 "LPC55S0x/LPC550x Power Profiles/Power Control API" on page 302.
- Configure the pin functions of any Micro-tick timer capture pins that will be used via IOCON, see Chapter 15 "LPC55S0x/LPC550x I/O Pin Configuration (IOCON)".
- Enable 1MHz clock by programming bit 4 in Syscon CLOCK_CTRL register. See Section 4.5.69 "Clock control".

25.4 General description

Figure 82 shows a conceptual view of the Micro-tick timer.

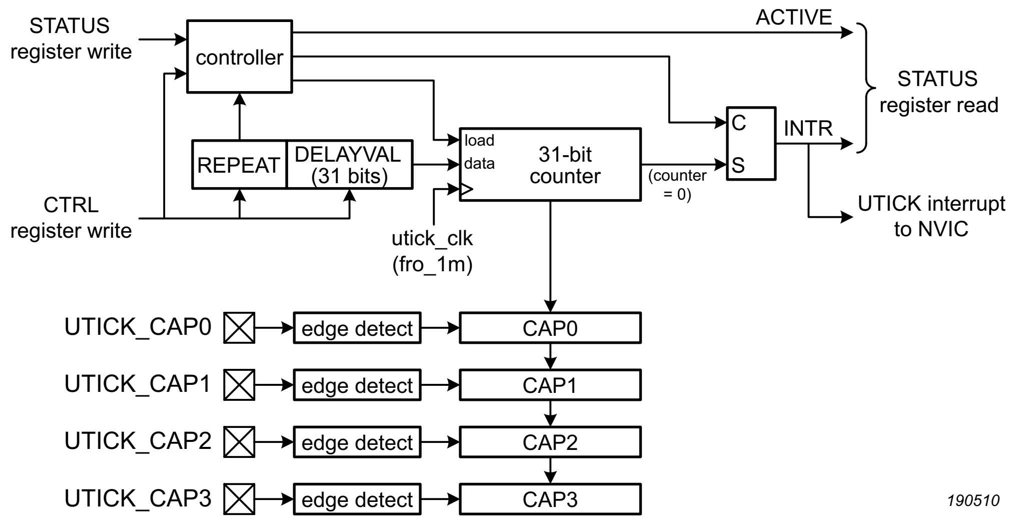


Fig 82. MIcro-tick timer block diagram

25.5 Pin description

Table 524 gives a summary of pins related to the Micro-tick Timer.

Table 524. 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,	Input	Capture inputs. The selected transition on a capture pin can be configured to load the related CAP register with the value of counter.

25.6 Register description

The Micro-tick Timer contains the registers shown in Table 525. 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 525. Register overview: Micro-tick Timer (base address = 0x5000 E000)

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	Control register.	0	25.6.1
STAT	R/W	0x004	Status register.	0	25.6.2
CFG	R/W	0x008	Capture configuration register.	0	25.6.3
CAPCLR	WO	0x00C	Capture clear register.	NA	25.6.4
CAP0	RO	0x010	Capture register 0.	0	25.6.5
CAP1	RO	0x014	Capture register 1.	0	25.6.5
CAP2	RO	0x018	Capture register 2.	0	25.6.5
CAP3	RO	0x01C	Capture register 3.	0	25.6.5

25.6.1 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 526. 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

25.6.2 Status register

This register provides status for the Micro-tick Timer.

Table 527. 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	-

25.6.3 Capture configuration register

This register allows enabling Micro-tick capture functions and selects the polarity of the capture triggers.

Table 528. 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	-

25.6.4 Capture clear register

This read-only register allows clearing previous capture values, allowing new captures to take place.

Table 529. 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	-

25.6.5 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 530. 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

UM11424

Chapter 26: LPC55S0x/LPC550x Multi-Rate Timer

Rev. 1.5 — 21 December 2023 User manual

26.1 How to read this chapter

The MRT (Multi-Rate Timer) is available on all LPC55S0x/LPC550x devices.

26.2 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.

26.3 Basic configuration

To configure the MRT, follow these general steps:

- 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.

26.4 Pin description

The MRT is not associated with any device pins.

26.5 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 26.5.1 “Repeat interrupt mode”
- One-shot interrupt mode. See Section 26.5.2 “One-shot interrupt mode”
- One-shot stall mode. See Section 26.5.3 “One-shot stall mode”

The modes for each timer are set in the timer’s control register. See Table 534.

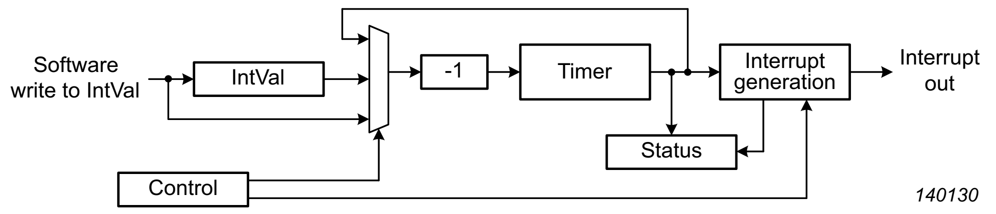


Fig 83. MRT block diagram

26.5.1 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 an 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.

26.5.2 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.

26.5.3 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.

26.6 Register description

The reset values in Table 531 are POR reset values.

Table 531. 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	26.6.1
TIMER0	RO	0x4	MRT0 timer. This register reads the value of the down-counter.	0xFF FFFF	26.6.2
CTRL0	R/W	0x8	MRT0 control. This register controls the MRT0 modes.	0	26.6.3
STAT0	R/W	0xC	MRT0 status.	0	26.6.4
MRT Timer 1 registers					
INTVAL1	R/W	0x10	MRT1 time interval value. This value is loaded into the TIMER1 register.	0	26.6.1
TIMER1	RO	0x14	MRT1 timer. This register reads the value of the down-counter.	0xFF FFFF	26.6.2
CTRL1	R/W	0x18	MRT1 control. This register controls the MRT0 modes.	0	26.6.3
STAT1	R/W	0x1C	MRT1 Status.	0	26.6.4
MRT Timer 2 registers					
INTVAL2	R/W	0x20	MRT2 time interval value. This value is loaded into the TIMER2 register.	0	26.6.1
TIMER2	RO	0x24	MRT2 timer. This register reads the value of the down-counter.	0xFF FFFF	26.6.2
CTRL2	R/W	0x28	MRT2 control. This register controls the MRT0 modes.	0	26.6.3
STAT2	R/W	0x2C	MRT2 status.	0	26.6.4
MRT Timer 3 registers					
INTVAL3	R/W	0x30	MRT3 time interval value. This value is loaded into the TIMER3 register.	0	26.6.1
TIMER3	RO	0x34	MRT3 timer. This register reads the value of the down-counter.	0xFF FFFF	26.6.2
CTRL3	R/W	0x38	MRT3 control. This register controls the MRT0 modes.	0	26.6.3
STAT3	R/W	0x3C	MRT3 status.	0	26.6.4
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	26.6.5
IDLE_CH	RO	0xF4	Idle channel. This register returns the number of the first idle channel.	0	26.6.6
IRQ_FLAG	R/W	0xF8	Global interrupt flag.	0	26.6.7

26.6.1 Time interval register

This register contains the MRT load value and controls how the timer is reloaded. The load value is IVALUE -1.

Table 532. 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.	

26.6.2 Timer register

The timer register holds the current timer value. This register is read-only.

Table 533. 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.	-

26.6.3 Control register

The control register configures the mode for each MRT and enables the interrupt.

Table 534. 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 534. 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.	-

26.6.4 Status register

This register indicates the status of each MRT.

Table 535. 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. See for details of the two operating modes.	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.	-

26.6.5 Module configuration register

The MODCFG register provides the configuration (number of channels and timer width) for this MRT. See for details.

Table 536. 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 536. 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.	

26.6.6 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 537. 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.	-

26.6.7 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 538. 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 TIMER2. See description of channel 0.	0
3	GFLAG3	-	Monitors the interrupt flag of TIMER3. See description of channel 0.	0
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	0

Chapter 27: LPC55S0x/LPC550x Real-Time Clock (RTC)

27.1 How to read this chapter

The RTC is available on all LPC55S0x/LPC550x devices.

27.2 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.

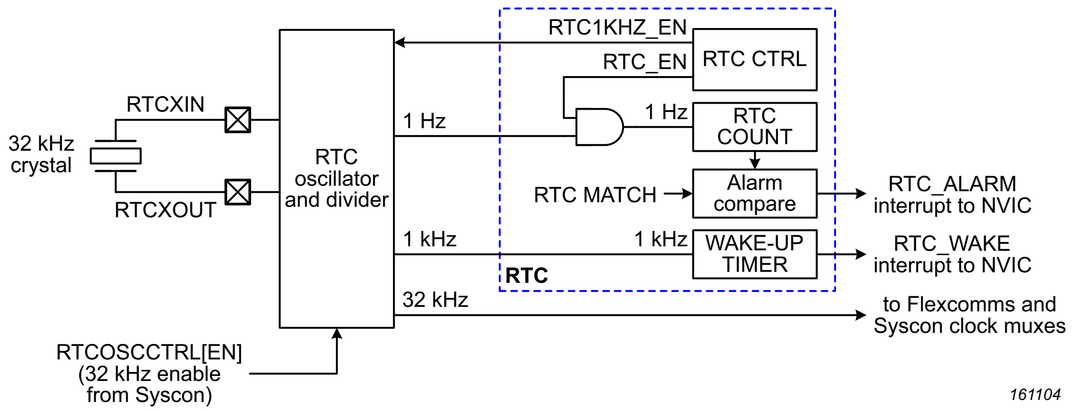
27.3 Basic configuration

Configure the RTC as follows:

- Use the AHBCLKCTRL0 register, see Section 4.5.16 “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 541. 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 “LPC55S0x/LPC550x 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 541.
- 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 38. 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.


RTC1KHZ_EN
RTC_EN
RTC CTRL
1 Hz
RTC COUNT
RTCXIN
32 kHz crystal
RTCXOUT
RTC oscillator and divider
1 Hz
RTC MATCH
Alarm compare
RTC_ALARM interrupt to NVIC
1 kHz
1 kHz
WAKE-UP TIMER
RTC_WAKE interrupt to NVIC
RTC
32 kHz
to Flexcomms and Syscon clock muxes
RTCOSCCTRL[EN] (32 kHz enable from Syscon)
161104

Note: Although the 32 kHz crystal is available, the 32 kHz FRO is used as the clock source in the default configuration.

Fig 84. RTC clocking and block diagram

27.3.1 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.

27.4 General description

27.4.1 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.

27.4.2 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.

27.4.3 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.

27.4.4 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.

27.4.5 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.

27.5 Pin description

Table 539 gives a summary of pins related to the RTC.

Table 539. RTC pin description

Pin	Type	Description
RTCXIN	Input	RTC oscillator input.
RTCXOUT	Output	RTC oscillator output.

27.6 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 540. 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	27.6.1
MATCH	R/W	0x04	RTC match.	0xFFFF FFFF	0xFFFF FFFF	27.6.2
COUNT	R/W	0x08	RTC counter.	0	0	27.6.3
WAKE	R/W	0x0C	High-resolution/wake-up timer control.	0	0	27.6.4
SUBSEC	RO	0x10	RTC sub-second counter.	0	0	27.6.5
GPREG0	R/W	0x40	General purpose register 0.	0	0	27.6.6
GPREG1	R/W	0x44	General purpose register 1.	0	0	27.6.6
GPREG2	R/W	0x48	General purpose register 2.	0	0	27.6.6
GPREG3	R/W	0x4C	General purpose register 3.	0	0	27.6.6
GPREG4	R/W	0x50	General purpose register 4.	0	0	27.6.6
GPREG5	R/W	0x54	General purpose register 5.	0	0	27.6.6
GPREG6	R/W	0x58	General purpose register 6.	0	0	27.6.6
GPREG7	R/W	0x5C	General purpose register 7.	0	0	27.6.6

27.6.1 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 541. 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 541. 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 541. 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.	-

27.6.2 RTC match register

Table 542. 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

27.6.3 RTC counter register

Table 543. 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

27.6.4 RTC high-resolution/wake-up register

Table 544. 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.	-

27.6.5 RTC sub-second counter

Table 545. 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.	-

27.6.6 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 546. 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

UM11424

Chapter 28: LPC55S0x/LPC550x System Tick Timer

Rev. 1.5 — 21 December 2023 User manual

28.1 How to read this chapter

There are two system tick timers (SysTick timer), one is secured and the other is non-secured.

Each tick timer has its own calibration provided by the Syscon.

28.2 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.

28.3 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).

28.4 General description

See Figure 85 for the block diagram of the SysTick timer.

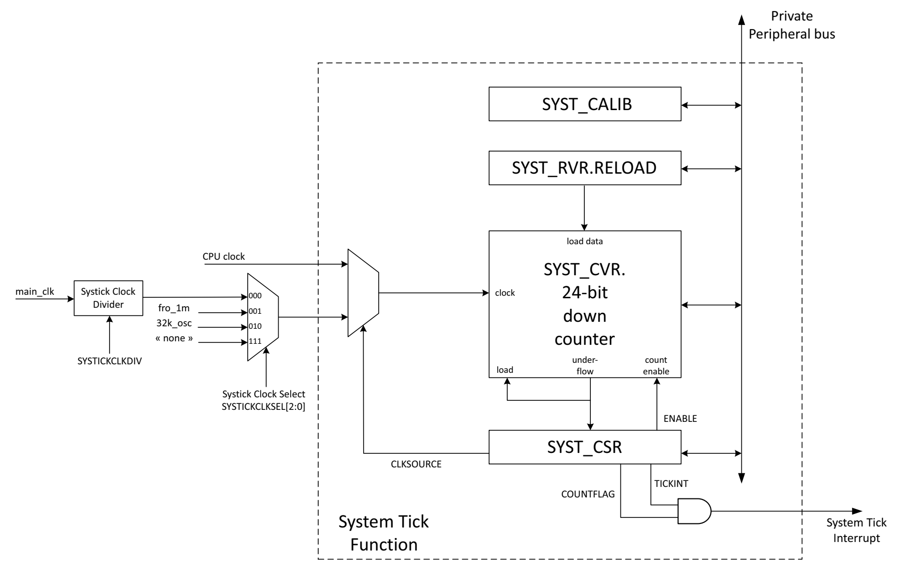


Fig 85. System tick timer block diagram

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.

28.5 Register description

The systick timer registers are located on the private peripheral bus of each CPU (see Figure 2).

Table 547. 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	28.5.1
SYST_RVR	R/W	0x014	System Timer Reload value register	0	28.5.2
SYST_CVR	R/W	0x018	System Timer Current value register	0	28.5.3
SYST_CALIB	RO	0x01C	System Timer Calibration value register	0	28.5.4

[1] Reading an writing have specific side effects, see detailed register descriptions for more details.

28.5.1 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 548. 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. **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

28.5.2 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 the timer initialization. The SYST_CALIB register may be read and used as the value for the SYST_RVR register if the CPU is running at the frequency intended for use with the SYST_CALIB value.

Table 549. 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

28.5.3 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 550. 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

28.5.4 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) registers in the system configuration block. See Table 39.

Table 551. 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 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 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 register in Syscon. When 0, a separate reference clock is available. When 1, a separate reference clock is not available.	0

28.6 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 2. 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 may 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).The SysTick 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 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 registers, using the SKEW and NOREF fields.

28.7 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 4.5.43 "SYSTICK clock divider register 0" and Section 4.5.25 "System Tick Timer for CPU0 source select".

Use DIV of the SYSTICKCLKDIV0 (CPU0) 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

UM11424

Chapter 29: LPC55S0x/LPC550x Windowed Watchdog Timer (WWDT)

Rev. 1.5 — 21 December 2023 **User manual**

29.1 How to read this chapter

The watchdog timer is available on all LPC55S0x/LPC550x devices.

29.2 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 86 that can be divided by WDT clock divider register value in SYSCON module. See Table 104. 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 “LPC55S0x/LPC550x Analog control”.
- The watchdog timer can be configured to run in deep-sleep mode.
- Debug mode.

29.3 Basic configuration

Configuration of the WWDT is accomplished as follows:

- Configure WDTCLKDIV register. See Table 104. 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.

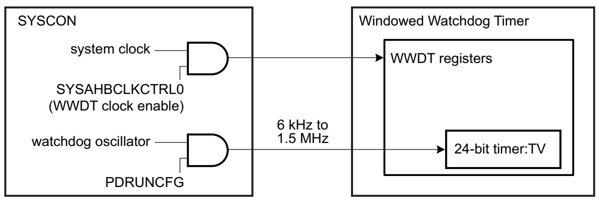


Fig 86. WWDT clocking

29.4 Pin description

The WWDT has no external pins.

29.5 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 29.3 "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 CPU will be reset, loading the stack pointer and program counter from the vector table as for an external reset. The watchdog time-out flag (WDTOF) can be examined to determine if the watchdog has caused the reset condition. The WDTOF flag must be cleared by software.

When the watchdog timer is configured to generate a warning interrupt, the interrupt will occur when the counter is no longer greater than the value defined by the WARNINT register.

29.5.1 Block diagram

The block diagram of the watchdog is shown in the Figure 87. The synchronization logic (APB bus clock to WDCLK) is not shown in the block diagram.

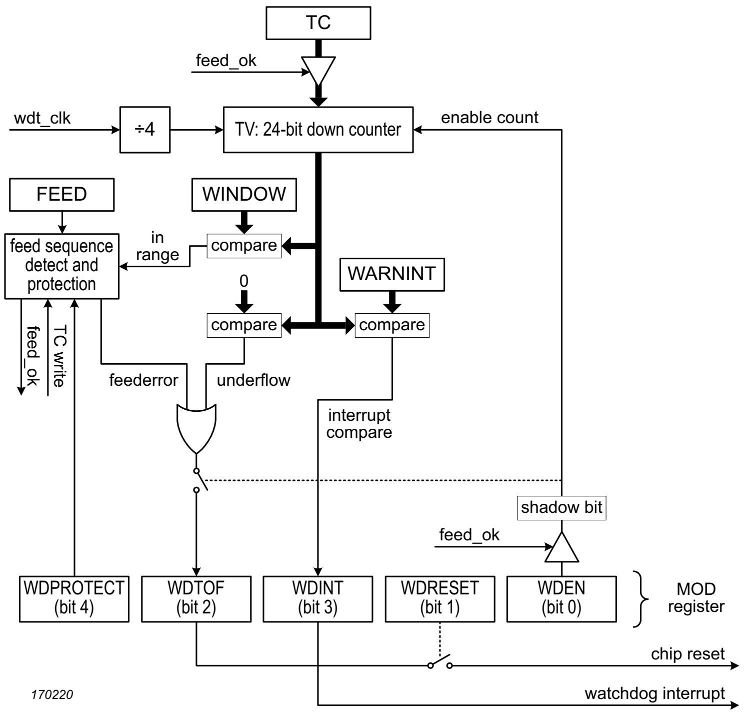


Fig 87. Windowed watchdog timer block diagram

29.5.2 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 87. 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.

29.5.3 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.

29.5.3.1 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.

29.6 Register description

The watchdog timer contains the registers shown in Table 552.

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

Table 552. 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	29.6.1
TC	R/W	0x004	Watchdog timer constant. This 24-bit register determines the time-out value.	0xFF	29.6.2
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	29.6.3
TV	RO	0x00C	Watchdog timer value. This 24-bit register reads out the current value of the watchdog timer.	0xFF	29.6.4
-	-	0x010	Reserved	-	-
WARNINT	R/W	0x014	Watchdog warning interrupt compare value.	0	29.6.5
WINDOW	R/W	0x018	Watchdog window compare value.	0xFF FFFF	29.6.6

29.6.1 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 553. 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 553. 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, and has been enabled using the POWER_EnterDeepSleep() API, the device will wake up.

Table 554. 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.

29.6.2 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 555. 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.	-

29.6.3 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 556. 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.	-

29.6.4 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 557. 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.	-

29.6.5 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 558. 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.	-

29.6.6 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.

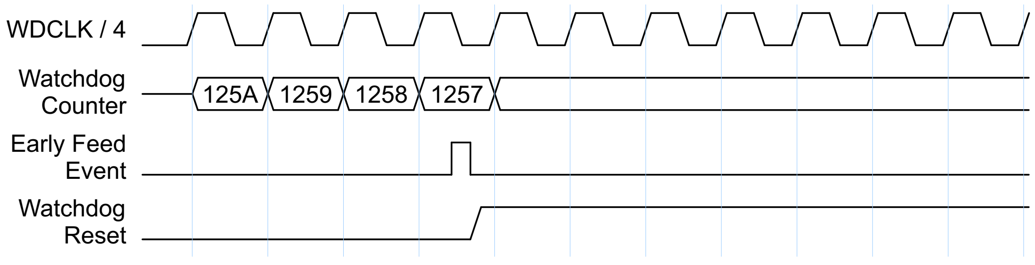
WINDOW resets to the maximum possible TV value, so windowing is not in effect.

Table 559. 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.	-

29.7 Functional description

The following figures illustrate several aspects of watchdog timer operation.



Conditions:
WINDOW = 0x1200
WARNINT = 0x3FF
TC = 0x2000

Fig 88. Early watchdog feed with windowed mode enabled

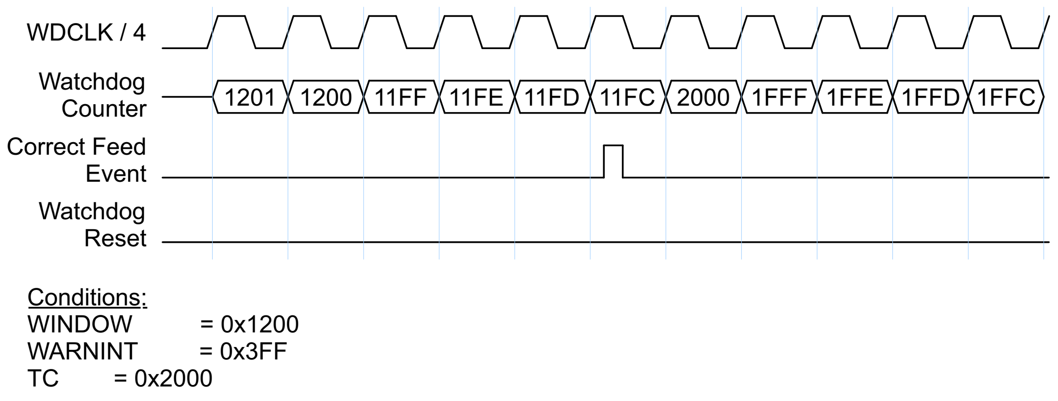


Fig 89. Correct watchdog feed with windowed mode enabled

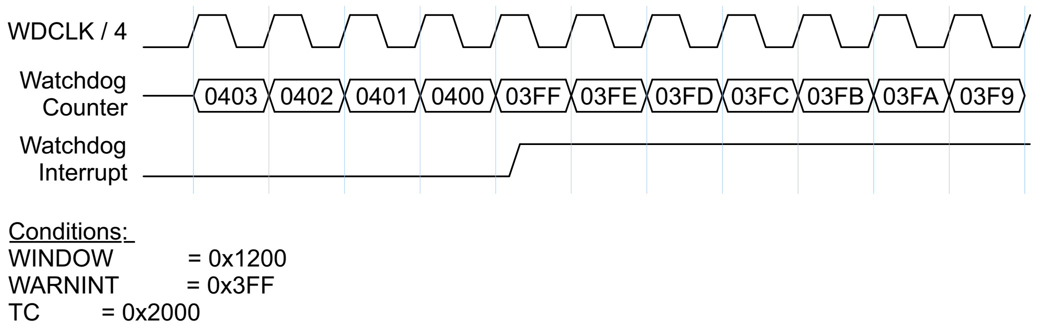


Fig 90. Watchdog warning interrupt

Chapter 30: LPC55S0x/LPC550x Code Watchdog Timer

30.1 How to read this chapter

The Code Watchdog Timer (CWT) is available on all LPC55S0x/LPC550x devices.

30.2 Introduction

The CWT provides two primary mechanisms for detecting code flow integrity as follows.

Secure Counter (SEC_CNT): Detects altered software in the execution flow.

- This counter (i.e., an accumulator) is loaded with an initial value and then the runtime software issues ADD & SUB commands to increment/decrement the counter.
- Periodically, a secure counter value check is initiated by passing the expected value to the CWD using the STOP and RESTART commands.
- If a mismatch is detected between the Secure Counter and the value passed to it, the execution flow has potentially been altered by a side channel attack or some other suspicious activity.

Instruction Timer (INST_TIMER): Places a hard upper-limit on the interval between checks of the secure counter.

- The START command loads the internal decremental counter. Before the counter generates an underflow (reaches 0), a STOP or RESTART command must be executed to force a secure counter check.

30.2.1 Secure Counter

The Secure Counter, is a 32-bit accumulating register that holds a dynamically changing value that can periodically be evaluated to determine if a program is executing in an expected manner. If a mismatch is detected, a FAULT is generated.

30.2.2 Instruction timer

The Instruction Timer is a 32-bit count-down timer that is used by an application to set the number of instructions that are expected to be executed and can thus detect cases where the counter is reset (set to 0) which may be an indication that unauthorized instructions are being executed. The application programmer pre-loads the Instruction Timer with slightly more than the number of instructions in the next execution sequence. The Instruction Timer counts off the instructions as they are executed (clocks) and if the number is exhausted before the CWT is serviced, a FAULT is generated. The Instruction Timer can be programmed to either pause, or keep running while the interrupt service routines are executing.

30.3 Architectural Design

The following block diagram shows the architectural components of the CWT.

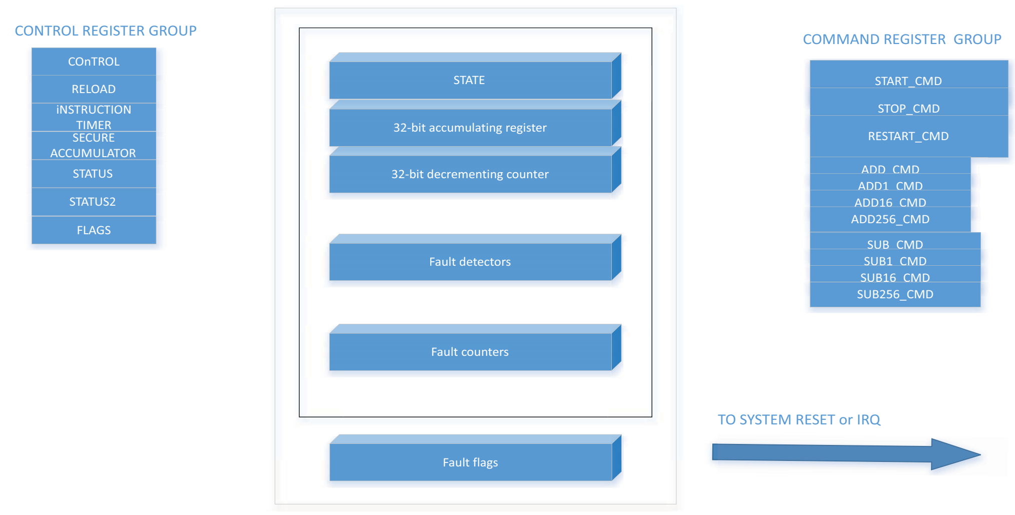


Fig 91. Block diagram of the Code Watchdog Timer

30.3.1 The CONTROL group of registers

The CONTORL group of registers perform the following functions:

- Configures the module through writes to registers in the control group.
- Reads the Instruction Timer current value and module status in this group.
- The Secure Counter is write-only.

30.3.2 The COMMAND group of registers

With writes to the COMMAND group, the running module is updated at specific points during the execution flow of the application. Through timely and accurate updates, the generation of FAULTs is prevented.

30.3.3 STATE

The STATE registers perform the following functions:

- The CWD has two legal states: IDLE and ACTIVE.
- Internally, the two states are encoded by four bits, with only two of the possible sixteen combinations permissible.
- After any reset, (including a reset generated by the CWD itself), the module will be in IDLE state.
- A correct sequence of CPU writes to the CONTROL group, followed by a START command, changes the state to ACTIVE.

- Once ACTIVE, a correctly formulated STOP COMMAND will change the state back to IDLE.

30.3.4 FAULT detectors

There are five types of faults that are detectable, where each can be individually controlled to generate either a system reset, an interrupt, or left idle.

30.3.5 FAULT counters

Each FAULT type has an associated counter, which increments when the fault is detected. The counters retain their value through system reset, and are cleared by POR.

30.3.6 FAULT flags

- Each fault type has an associated flag, that is set when the fault is detected.
- The flags retain their value through system reset.
- They are cleared by POR, or by software.

30.4 Operation

30.4.1 During Code Development Debug

Following are suggestions for facilitating the code development and debug cycle:

1. Use interrupt-on-fault (instead of reset) to maintain control while dialing in the timing and the counting values to be used.
2. Direct-write the bits in the flags register to trigger faults, with the same result as hardware-triggered faults.
3. Use the DEBUG_HALT_CTRL field to pause the Instruction Timer during the pausing of a debug session.

30.4.2 Model use cases

There is generally a strict sequence with which the CWD is configured, activated and serviced as follows:

1. Write an Inst. Tim. Rel. and Val., corresponding to the current code section, to the RELOAD register.
2. Write a control word to the CONTROL register, where each fault type is Enabled (or not) to generate a reset, and lock the lock field. The Instruction Timer may be kept running at all times, or only during non-IRQ execution (paused during interrupt processing) based on the value that is written to the IRQ_PAUSE field.
3. Activate the module by writing to the START command register. The initial value for the Sec. Counter is the value written. The Instruction Timer immediately starts decrementing from the RELOAD value on every clock cycle.
4. At strategically chosen way points in the application's code flow, update the Sec. Cnt. by issuing ADD or SUB commands.

5. When the way point that corresponds to the number of executed instructions represented by the RELOAD value (the end of the current code section) is reached, write the expected value of the S.C. to the STOP command register. Assuming that the written value compares exactly to the contents of the S.C., the Instruction Timer stops and the process can begin again for the next code section, by repeating steps 1 thru 5.

Another example of a configuration and start procedure would proceed as follows:

1. Write the Inst. Tim. Rel. Val. to the RELOAD register. It is a good idea to write a very large number to provide a buffer for large values.
2. Write a control word to the CONTROL register, where each fault type is enabled (or not) to generate a reset and then lock the lock field.
3. Activate the module by Writing to the START command register. The initial value for the Secure counter is the value written. The Inst. Tim. immediately starts decrementing from the RELOAD value on every clock.
4. At strategically chosen way points in the application's code flow, update the Secure counter by issuing ADD or SUB commands.
5. At all times within the execution flow, the application must be aware of the instruction counter's approach towards zero. The Instruction Timer may require regular reading to determine the expended time. Software needs to service the Dawg before the TIMER reaches 0, by writing the Sec. Cntr. expected value to the RESTART command register. Assuming the value written compares exactly to the contents of the Sec. Accum, the Instruction Timer is reloaded with the value in the RELOAD register as it decrements towards 0.

30.5 Details on faults, flags and counters

30.5.1 Fault types

There are six types of faults that are detectable (TIMEOUT, MISCOMPARE, SEQUENCE, CONTROL, STATE, ADDRESS). Each type can be individually controlled to generate either a system reset, an interrupt, or nothing. It should be noted that illegal values in the various bitfields of the CONTROL register itself can only enabled to generate either a system reset, or nothing.

Interrupts and resets can both be enabled simultaneously, but each fault type can only contribute to one or the other, not both. Interrupts are available as an alternative to system reset generation, for all fault types (except the CONTROL fault), primarily to facilitate code development and debug operations. In the final application, interrupts can be enabled for some faults, and reset can be enabled for other faults (or disabled completely), at the risk of reduced security.

30.5.1.1 TIMEOUT

A TIMEOUT fault will be raised when the Instruction Timer times out (i.e., reaches '0').

30.5.1.2 MISCOMPARE

A MISCOMPARE fault will be raised when either a STOP or a RESET command is issued, and the value passed in the instruction does not match (i.e., miscompares with) the content of the SEC. CNTR.

30.5.1.3 SEQUENCE

A SEQUENCE fault will be raised when the prescribed sequence of interactions between SW and CWDog is violated.

30.5.1.4 CONTROL

A CONTROL fault will be raised if any of the CONTROL register's fields contain an illegal value.

30.5.1.5 STATE

A STATE fault will be raised if the internal state machine ends up in one of the fourteen illegal combinations that can be encoded by STATE's four-bit encoding.

30.5.1.6 ADDRESS

An ADDRESS fault will be raised when any address in the module's register space, which is not legally defined, is accessed by the CPU.

30.5.2 Flags

The CWD has one flag for each fault type which are accessible through the FLAGS register. A flag will be set whenever its associated fault is detected, and can be cleared by SW. The flags themselves (when enabled) generate the module's system reset or interrupt outputs.

The flags retain their value through a system reset (including one generated by the CWD), but are reset by a POR. Using this mechanism, SW can easily answer the 'How did I get here?' question by reading the FLAGS register after any reset.

To facilitate testing and code development, the flags can be written directly when the module is not locked. When the module is locked, the flags can be cleared by writing '1' to their bit positions.

The exception is the PORF flag, which is purely for status, and contributes to neither system reset nor interrupt. It is set to '1' after POR, and can be cleared by SW with a write of '1' to its bit position, but cannot be set by software.

Table 560. FLAGS summary

Name	PORF	ADDR	STATE	CONTROL	SEQUENCE	MISCOMPARE	TIMEOUT
Bit	16	5	4	3	2	1	0
Reset value after POR?	1	0	0	0	0	0	0
Writeable when unlocked?	N	Y	Y	Y	Y	Y	Y
W-12-C when locked?	Y	Y	Y	Y	Y	Y	Y

30.5.3 Fault counters

Each fault type has a counter that increments when a fault of the specified type is detected. The counters are cleared by POR, but not by a system reset. Thus, statistics can be built up over many code watchdog resets to reveal the behavior patterns of a specific type of attack.

30.5.4 Fault exposure

The CWD supports the following faults:

- RELOAD must be written before the START command is issued. Failure to do so generates an illegal sequence fault.
- RELOAD can only be written in IDLE state. Attempt to write to RELOAD in ACTIVE state generates an illegal sequence fault.
- START can only be written in IDLE state, and only if RELOAD has first been written. Attempt to write to START in ACTIVE state generates an illegal sequence fault.
- The TIMEOUT fault is generated one clock before the Instruction Timer reaches '0.
- The MISCOMPARE fault is generated when a write to the 4K Byte address space occurs and:
  – The STOP register is addressed and the current secure counter value does not match the write data.
  – The RESTART register is addressed and the current secure counter value does not match the write data.
- An illegal SEQUENCE fault is generated when a write to the 4K Byte address space occurs and:
  – The START register is addressed and the RELOAD register has not been written.
  – The register addressed is any register in the command group other than START, and the state machine is in IDLE state.
  – The address written is any address not in the COMMAND group, or it is the START command address, and the state machine is in ACTIVE state.
- An Illegal CONTROL fault is generated when:
  – The LOCK_CTRL field is any value other than 01b or 10b.
  – The TIMEOUT_CTRL, MISCOMP_CTRL, SEQEUNCE_CTRL, CONTROL_CTRL, STATE_CTRL, or ADDRESS_CTRL fields contain any value other than 001b, 010b, or 100b.
  – The DEBUG_HALT_CTRL field contains any value other than 01b, or 10b.
  – The INTERRUPT_HALT_CTRL field contains any value other than 01b, or 10b.
- An Illegal STATE fault is generated at any time that the STATE machine contains any value other than 5h or Ah.
- An Illegal ADDRESS fault is generated when:
  – A read access to any address outside the CONTROL register group within the 4K byte address space.
  – A write access to any address outside both the CONTROL and COMMAND register groups within the 4K byte address space.

30.6 Control and Status register descriptions

Following are descriptions for the register groups.

Table 561. Register overview: CDOG registers (base address 0x400A1000)

Name	Access	Offset	Description	Reset value	Section
CONTROL	RW	0x0	Controls attributes of the module, including those of CONTROL itself.	0x50092492	30.6.1
RELOAD	RW	0x4	Instruction Timer reload.	0xFFFFFFFF	30.6.2
INSTRUCTION_TIMER	RW	0x8	Instruction Timer.	0xFFFFFFFF	30.6.3
SECURE_COUNTER	RW	0xC	Secure counter.	0x0	30.6.5
STATUS	RO	0x10	Status register (1 of 2).	0x0	30.6.6
STATUS2	RO	0x14	Status register (2 of 2).	0x0	30.6.7
FLAGS	RW	0x18	Hardware flags.	0x0	30.6.8
PERSISTENT	RW	0x1C	Persistent data storage.	0x0	30.6.9
START	W	0x20	Write address for issuing the START command.	0x0	30.6.10
STOP	W	0x24	Write address for issuing the STOP command.	0x0	30.6.11
RESTART	W	0x28	Write address for issuing the RESTART command.	0x0	30.6.12
ADD	W	0x2C	Write address for issuing the ADD command.	0x0	30.6.13
ADD1	W	0x30	Write address for issuing the ADD1 command.	0x0	30.6.14
ADD16	W	0x34	Write address for issuing the ADD16 command.	0x0	30.6.15
ADD256	W	0x38	Write address for issuing the ADD256 command.	0x0	30.6.16
SUB	W	0x3C	Write address for issuing the SUB command.	0x0	30.6.17
SUB1	W	0x40	Write address for issuing the SUB1 command.	0x0	30.6.18
SUB16	W	0x44	Write address for issuing the SUB16 command.	0x0	30.6.19
SUB256	W	0x48	Write address for issuing the SUB256 command.	0x0	30.6.20

30.6.1 Control

The control fields, which constitute CONTROL, control all controllable attributes of the module, including those of CONTROL itself.

Table 562. CONTROL register description. (CONTROL, offset 0x0)

Bit	Symbol	Access	Description	Reset Value
1:0	LOCK_CTRL	RW	Lock control field.	0x2
			001 = Locked.	
			010 = Unlocked.	
4:2	TIMEOUT_CTRL	RW	TIMEOUT control.	0x4
			001 = Enable reset.	
			010 = Enable interrupt.	
			100 = Disable both when timeout occurs.	
7:5	MISCOMPARE_CTRL	RW	MISCOMPARE control field.	0x4

Table 562. CONTROL register description. (CONTROL, offset 0x0) *…continued*

Bit	Symbol	Access	Description	Reset Value
			001 = Enable reset.	
			010 = Enable interrupt.	
			100 = Disable both when mis-compare fault occurs.	
10:8	SEQUENCE_CTRL	RW	SEQUENCE control field.	0x4
			001 = Enable reset.	
			010 = Enable interrupt.	
			100 = disable both when illegal sequence fault occurs.	
13:11	CONTROL_CTRL	RW	CONTROL control field.	0x4
			001 = Enable reset.	
			010 = Enable interrupt.	
			100 = disable both when illegal CONTROL fault occurs.	
16:14	STATE_CTRL	RW	STATE control field.	0x4
			001 = Enable reset.	
			010 = Enable interrupt.	
			100 = disable both when illegal state fault occurs.	
19:17	ADDRESS_CTRL	RW	ADDRESS control field.	0x4
			001 = Enable reset.	
			010 = Enable interrupt.	
			100 = disable both when address fault occurs.	
27:20	Un_Imps	RU	The un-imps are un-IMPs.	0x0
29:28	IRQ_PAUSE	RW	IRQ pause control field.	0x1
			001 = Instruction timer runs during interrupts.	
			010 =10b=Instruction timer pauses during interrupts.	
31:30	DEBUG_HALT_CTRL	RW	DEBUG_HALT control field.	0x1
			001 =run.	
			010 = Pause the instruction timer when CPU asserts debug HALT.	

30.6.2 Reload

The RELOAD register contains the value from which the Instruction Timer will count down when a valid START or RESTART command is executed. This register can only be written to when the state machine is in an IDLE state and must be written before issuing a START command.

Table 563. (instruction timer reload value) 0x4Reset value = 0xFFFFFFFF

Bit	Symbol	Access	Description	Reset Value
31:0	RLOAD	RW	Instruction Timer counts down when a valid START or RESTART is issued.	0xFFFFFFFF

30.6.3 Instruction timer

The Instruction timer is a 32-bit count-down timer that sets the number of instructions that are expected to be executed which can then be monitored by an application for suspicious activity. The current value of the timer can be read from this address. Not writable by the CPU.

Table 564. The Instruction timer (INSTRUCTION_TIMER, offset 0x8)

Bit	Symbol	Access	Description	Reset Value
31:0	INSTIM	RW	Instruction Timer 32-bit value.	0xFFFFFFFF

30.6.4 Instruction timer reload

The Instruction timer reload register reloads the timer based on the supplied value.

Table 565. Instruction timer reload (RELOAD, offset 0x4)

Bit	Symbol	Access	Description	Reset Value
31:0	RLOAD	RW	Instruction timer reload value.	0xFFFFFFFF

30.6.5 Secure counter

The Secure counter, (also known as SEC_CNT), supplies the start value for the counter.

Table 566. Secure counter (SECURE_COUNTER, offset 0xC)

Bit	Symbol	Access	Description	Reset Value
31:0	SEC_CNT	RW	Secure counter.	0x0

30.6.6 Status register 1

Following are descriptions for Status register 1.

Table 567. Status register (1 of 2) (STATUS, offset 0x10)

Bit	Symbol	Access	Description	Reset Value
7:0	NUMTOF	RO	Number of Timeout Faults.	0x0
15:8	NUMMISCOMPF	RO	Number of Miscompare Faults.	0x0
23:16	NUMILSEQF	RO	Number of illegal sequence faults.	0x0
27:24	uN_iMps	RU	Reserved.	0x0
31:28	CURST	RO	Current state.	0x0

30.6.7 Status register 2

Following are descriptions for Status register 2.

Table 568. STATUS register (2 of 2) (STATUS2, offset 0x14)

Bit	Symbol	Access	Description	Reset Value
7:0	NUMCNTF	RO	Number (of) control faults.	0x0
15:8	NUMILLSTF	RO	Number (of) state faults.	0x0
23:16	NUMILLA	RO	Number of (illegal) address faults.	0x0
31:24	un_imPs	RU	Not implemented.	0x0

30.6.8 Flags

The flags retain their value through a system reset (including one generated by the CWD), but are reset by a POR.

Table 569. Hardware flags (FLAGS, offset 0x18)

Bit	Symbol	Access	Value	Description	Reset Value
0	TO_FLAG	RW		When unlocked, write-direct. When locked, WR-1-to-clear. Set by hardware upon detection of a time-out fault. When '1' this bit will cause a reset or an interrupt if enabled (see CONTROL).	0x0
1	MISCOM_FLAG	RW		When unlocked, write-direct. When locked, WR-1-to-clear. Set by hardware upon detection of a miscompare fault. When '1', this bit will cause a reset or an interrupt if enabled (see CONTROL).	0x0
2	SEQ_FLAG	RW		When unlocked, write-direct. When locked, WR-1-to-clear. Set by hardware upon detection of an illegal sequence_fault. When '1', this bit will cause a reset or an interrupt if enabled (see CONTROL).	0x0
3	CNT_FLAG	RW		When unlocked, write-direct. When locked, WR-1-to-clear. Set by hardware upon detection of an illegal control_fault. When '1', this bit will cause a reset (no interrupt possibility) if enabled (see CONTROL).	0x0

Table 569. Hardware flags (FLAGS, offset 0x18)

Bit	Symbol	Access	Value	Description	Reset Value
4	STATE_FLAG	RW		When unlocked, write-direct. When locked, WR-1-to-clear. Set by hardware upon detection of an illegal state_fault. When '1', this bit will cause a reset or an interrupt if enabled (see CONTROL).	0x0
5	ADDR_FLAG	RW		When unlocked, write-direct. When locked, WR-1-to-clear. Set by hardware upon detection of an illegal address_fault. When '1', this bit will cause a reset or an interrupt if enabled (see CONTROL).	0x0
15:6	-	RU		Reserved.	0x0
16	POR_FLAG	RW		When unlocked, write-direct. When locked, WR-1-to-clear. Always set by POR.	0x0
31:17	-	RU		Reserved.	0x0

30.6.9 Persistent

The Persistent data storage specifies the location of the storage space used by CWD.

Table 570. Persistent (Ad. Hoc., quasi-NV) data storage (PERSISTENT, offset 0x1C)

Bit	Symbol	Access	Description	Reset Value
31:0	PERSIS	RW	Thirty-two scratch-pad bits are available to the application for storing information that persists through resets generated by the Code Watchdog Timer. The value written will be unchanged after any reset other than a POR reset.	0x0

30.6.10 Start

The 32-bit value written to this address is loaded into the SECURE COUNTER, the RELOAD value is loaded into the Instruction Timer, and the module enters an ACTIVE state in which the Instruction Timer counts down on every clock. This address can only be written during IDLE state, and only after writing to the RELOAD register. An illegal sequence fault is generated if this specified sequence is not followed.

Table 571. Write address for issuing the START command. (START, offset 0x20)

Bit	Symbol	Access	Description	Reset Value
31:0	STRT	WO	Address of START command access.	0x0

30.6.11 Stop

The 32-bit value written to this address is loaded into the SECURE COUNTER, the RELOAD value is loaded into the Instruction Timer, and the module enters ACTIVE state in which the Instruction Timer counts down on every clock. This address can only be written during IDLE state, and only after writing to the RELOAD register. An illegal sequence fault is generated if this specified sequence is not followed.

Table 572. Write address for issuing the STOP command. (STOP, offset 0x24)

Bit	Symbol	Access	Description	Reset Value
31:0	STP	WO	Address of STOP command.	0x0

30.6.12 Restart

The 32-bit value written to this address is compared to the current value of the SECURE COUNTER. If the values match, the Instruction Timer is reloaded with the contents of the RELOAD register and proceeds counting down. The module remains in ACTIVE state. This address can only be written during an ACTIVE state.

Table 573. Write address for issuing the RESTART command. (RESTART, offset 0x28)

Bit	Symbol	Access	Description	Reset Value
31:0	RSTRT	WO	Write address for issuing the RESTART command.	0x0

30.6.13 Add

The 32-bit value written to this address is added to the current value of the SECURE COUNTER. This address can only be written during an ACTIVE state.

Table 574. Write address for issuing the ADD command. (ADD, offset 0x2C)

Bit	Symbol	Access	Description	Reset Value
31:0	AD	WO	Address of ADD command.	0x0

30.6.14 Add 1

The value written to this address is ignored, and the value 1 is added to the current value of the SECURE COUNTER. This address can only be written during an ACTIVE state.

Table 575. Write address for issuing the ADD1 command. (ADD1, offset 0x3)

Bit	Symbol	Access	Description	Reset Value
31:0	AD1	WO	Address of ADD1 command.	0x0

30.6.15 Add 16

The value written to this address is ignored, and the value 16 is added to the current value of the SECURE COUNTER. This address can only be written during an ACTIVE state.

Table 576. Write address for issuing the ADD16 command. (ADD16, offset 0x34)

Bit	Symbol	Access	Description	Reset Value
31:0	AD16	WO	Address of ADD16 command.	0x0

30.6.16 Add 256

The value written to this address is ignored, and the value 256 is added to the current value of the SECURE COUNTER. This address can only be written during an ACTIVE state.

Table 577. Write address for issuing the ADD256 command. (ADD256, offset 0x38)

Bit	Symbol	Access	Description	Reset Value
31:0	AD256	WO	Address of ADD256 command.	0x0

30.6.17 Subtract

The 32-bit value written to this address is subtracted from the current value of the SECURE COUNTER. This address can only be written during an ACTIVE state.

Table 578. Write address for issuing the SUB command. (SUB, offset 0x3C)

Bit	Symbol	Access	Description	Reset Value
31:0	S0B	WO	Address of the SUB command.	0x0

30.6.18 Subtract 1

The value written to this address is ignored, and the value 1 is subtracted from the current value of the SECURE COUNTER. This address can only be written during an ACTIVE state.

Table 579. Write address for issuing the SUB1 command. (SUB1, offset 0x40)

Bit	Symbol	Access	Description	Reset Value
31:0	S1B	WO	Address of the SUB1 command.	0x0

30.6.19 Subtract 16

The value written to this address is ignored, and the value 16 is subtracted from the current value of the SECURE COUNTER. This address can only be written during an ACTIVE state.

Table 580. Write address for issuing the SUB16 command. (SUB16, offset 0x44)

Bit	Symbol	Access	Description	Reset Value
31:0	SB16	WO	Address of SUB16 command.	0x0

30.6.20 Subtract 256

The value written to this address is ignored, and the value 256 is subtracted from the current value of the SECURE COUNTER. This address can only be written during an ACTIVE state.

Table 581. Write address for issuing the SUB256 command. (SUB256, offset 0x48)

Bit	Symbol	Access	Description	Reset Value
31:0	SB256	WO	Address of SUB256 command.	0x0

UM11424

Chapter 31: LPC55S0x/LPC550x OS Event Timer

Rev. 1.5 — 21 December 2023 User manual

31.1 How to read this chapter

The OS Event timer is available on all LPC55S0x/LPC550x 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 92 shows a conceptual view of the OS Event timer.

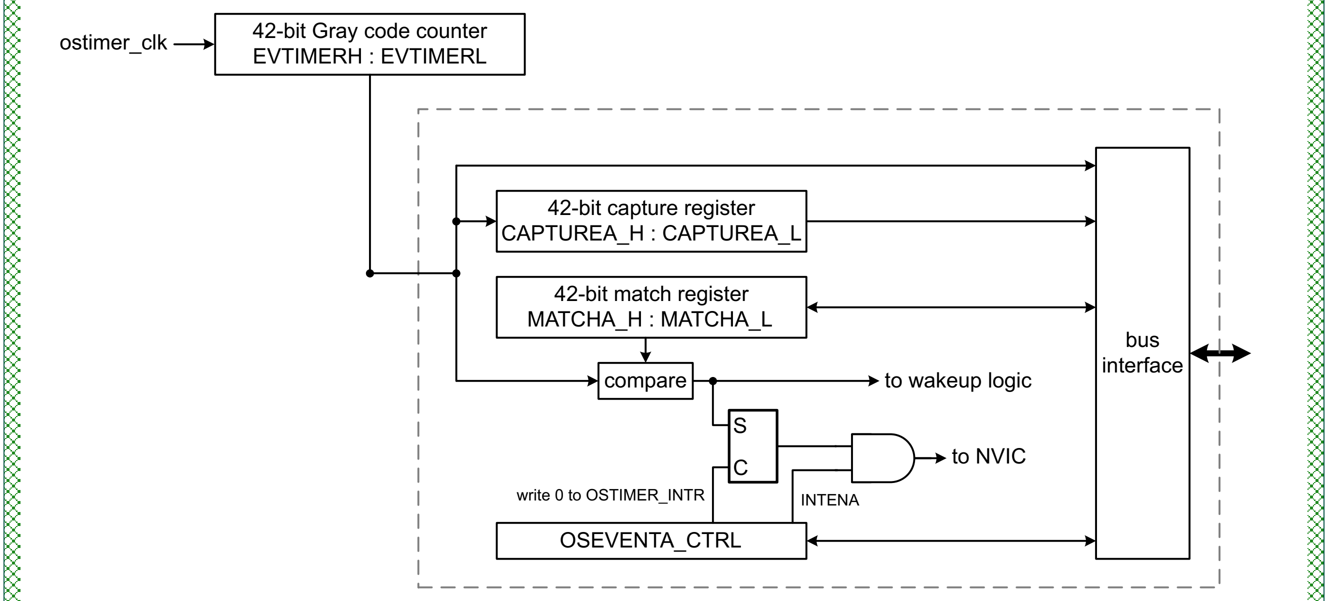


Fig 92. OS Event timer block diagram

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 582. Register overview: OS Event timer (base address = 0x4002D000)

Name	Access	Offset	Description	Reset value	Section
EVTIMERL	RO	0x0	Central EVTIMER low register.	0x00000000	31.6.1
EVTIMERH	RO	0x4	Central EVTIMER high register.	0x00000000	31.6.2
CAPTURE_L	RO	0x8	Capture low register.	0x00000000	31.6.3
CAPTURE_H	RO	0xC	Capture high register.	0x00000000	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 583. 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 that 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 584. 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 that there is only one EVTMER readable from all domains.	0x0
31:10	-	Reserved.	

31.6.3 Capture low register (CAPTURE_L)

This register resets only on system reset. Not effected by a software reset.

Table 585. Capture low register for CPU (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 586. Capture high register for CPU (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();").	0
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 587. Match low register for CPU (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 588. Match high register for CPU (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 (OSEVENT_CTRL)

This register resets only on system reset or a software reset.

Table 589. OS_EVENT TIMER control register for CPU (OSEVENT_CTRL, offset = 0x1C)

Bit	Symbol	Description	Reset value
0	OSTIMER_INTRFLAG	This bit is set when a match occurs between the central 64-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 and should be performed before a new match value is written into the MATCH_L/H registers.	0xFFFFFFFF
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.	0xFFFFFFFF
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 64-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.

Chapter 32: LPC55S0x/LPC550x Flexcomm Interface Serial Communication

32.1 How to read this chapter

Multiple Flexcomm Interfaces are available on all LPC55S0x/LPC550x 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^{2}C, including separate master, slave, and monitor functions.
- I^{2}S master or slave. Supports single I^{2}S channel.
- Data for USART, SPI, and I^{2}S traffic uses the Flexcomm Interface FIFO. The I^{2}C 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 4.5.17 “AHB clock control 1”.
2. Flexcomm Interface clock: Select a clock source for the related Flexcomm Interface. Options are shown in Figure 2. Also, see Section 4.5.39 “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 “LPC55S0x/LPC550x USARTs”, Chapter 35 “LPC55S0x/LPC550x Serial Peripheral Interfaces”, Chapter 33 “LPC55S0x/LPC550x I^{2}C-bus Interfaces”, and Chapter 37 “LPC55S0x/LPC550x I^{2}S interface”.

   Remark: The Flexcomm Interface (0 to 7) function clock frequency must not be higher than 48 MHz.

32.5 Architecture

The overall structure of one Flexcomm Interface is shown in Figure 93.

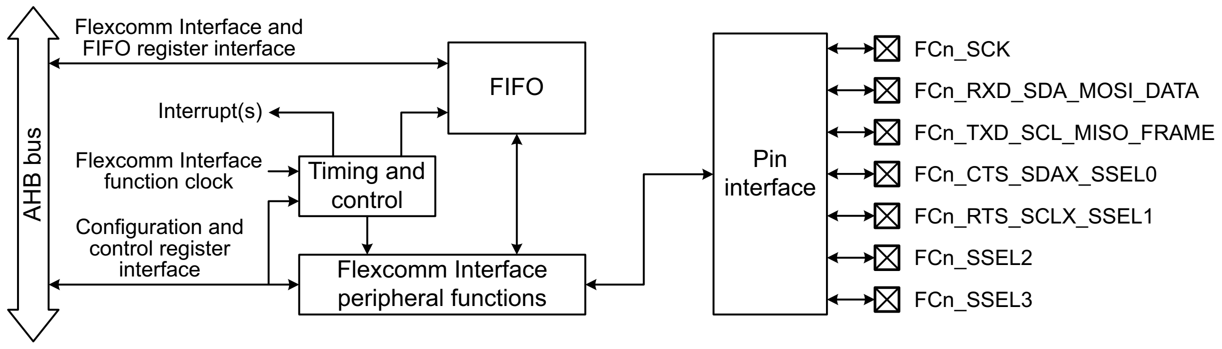

AHB bus
Flexcomm Interface and FIFO register interface
Interrupt(s)
Flexcomm Interface function clock
Configuration and control register interface
FIFO
Timing and control
Flexcomm Interface peripheral functions
Pin interface
FCn_SCK
FCn_RXD_SDA_MOSI_DATA
FCn_TXD_SCL_MISO_FRAME
FCn_CTS_SDAX_SSEL0
FCn_RTS_SCLX_SSEL1
FCn_SSEL2
FCn_SSEL3

Fig 93. Flexcomm Interface block diagram

32.5.1 Function Summary

LPC55S0x/LPC550x devices include Flexcomm Interfaces and functions as shown in Table 590. Specific part numbers and package variations may include a subset of this list.

Table 590. 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, special I²C pins available	Yes, 1 channel pair.
2	0x4008 8000	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, 1 channel pair.
6	0x4009 7000	Yes	Yes	Yes	Yes, 4 channel pair.
7	0x4009 8000	Yes	Yes	Yes	Yes, 4 channel pair.

Note: Flexcomm 7 does not exist on HVQFN48.

Remark: The FlexComm Interface8 has a different clock source selection than FlexComm Interface 0 to FlexComm Interface 7. See Chapter 4 "LPC55S0x/LPC550x 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 590.

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 590.

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 591. 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 (if present).
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 (if present).
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 (if present).
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^{2}C, and I^{2}S). The Flexcomm Interface registers that identify and configure the Flexcomm Interface are shown in Table 590.

The base addresses of all Flexcomm Interfaces may be found in Table 590.

Table 592. 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 593. 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^{2}C function selected.	
		0x4	I^{2}S transmit function selected.	
		0x5	I^{2}S 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^{2}C present indicator. This field is Read-only.	0
		0	This Flexcomm Interface does not include the I^{2}C function.	
		1	This Flexcomm Interface includes the I^{2}C function.	

Table 593. 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 594. 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

UM11424

Chapter 33: LPC55S0x/LPC550x I^{2}C-bus Interfaces

Rev. 1.5 — 21 December 2023 **User manual**

33.1 How to read this chapter

I^{2}C-bus functions are available on all LPC55S0x/LPC550x 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 I^{2}C pins).
  - High-speed mode, 3.4 Mbits/s as a slave only (on specific I^{2}C pins).
- Supports both Multi-master and Multi-master with slave functions.
- Multiple I^{2}C 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 I^{2}C 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 I^{2}C pins are fixed-pin functions and enabled through IOCON. See the IOCON settings in Table 595 and in Chapter 15 “LPC55S0x/LPC550x I/O Pin Configuration (IOCON)”.

Table 595. I^{2}C-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	I^{2}C serial clock.
SDA	I/O	FCn_RXD_SDA_MOSI_DATA or FCn_CTS_SDA_SSEL0	I^{2}C serial data.

33.4 Basic configuration

Configure the I^{2}C and related clocks as follows:

- If needed, use the PRESETCTRL1 or PRESETCTRL2 register, see Section 4.5.7 “Peripheral reset control 1” and Section 4.5.8 “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 4.5.17 “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 20 MHz when the maximum System Frequency chosen by the user is below or equal to 72 MHz.
- Above 100 MHz otherwise (when the maximum System Frequency chosen by the user is above 72 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 608.
- 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 I^{2}C interface, see Chapter 25 “LPC5500 Flexcomm Interface serial communication”.

Configure the I^{2}C as a master: set the MSTEN bit to 1 in the CFG register. See Table 601.

Write data to the slave:

1. Write the slave address with the $\overline{\text{RW}}$ bit set to 0 to the master data register MSTDAT. See Table 613.
2. Start the transmission by setting the MSTSTART bit to 1 in the master control register. See Table 610. The following happens:
   - The pending status is cleared and the I^{2}C-bus is busy.
   - The I^{2}C master sends the start bit and address with the $\overline{\text{RW}}$ 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 610. This step results in the following:
   - The pending status is cleared and the I^{2}C-bus is busy.
   - The I^{2}C 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 610.

Table 596. 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 I^{2}C 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 602.

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 613.

Start the transmission by setting the MSTSTART bit to 1 in the master control register. See Table 610. The following is accomplished:

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.

Waits for the pending status to be set (MSTPENDING = 1) by polling the STAT register.

Reads eight bits of data from the MSTDAT register.

Stops the transmission by setting the MSTSTOP bit to 1 in the master control register. See Table 610.

Table 597. 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 I^{2}C interface, see Chapter 32 “LPC55S0x/LPC550x Flexcomm Interface Serial Communication” Configure the I^{2}C as a slave with address x:

Set the SLVEN bit to 1 in the CFG register. See Table 602.

- Write the slave address x to the address 0 match register. See Table 615.

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 614.
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 614.

Acknowledge (*ack*) the data by setting SLVCONTINUE = 1 in the slave control register. See Table 613.

Table 598. 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 I^{2}C interface, Chapter 32 “LPC55S0x/LPC550x Flexcomm Interface Serial Communication” Configure the I^{2}C as a slave with address x:

- Set the SLVEN bit to 1 in the CFG register. See Table 601.
- Write the slave address x to the address 0 match register. See Table 615.

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 613.
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 614.

Continue the transaction by setting SLVCONTINUE = 1 in the slave control register. See Table 613.

Table 599. 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 I^{2}C for wake-up

In sleep mode, any activity on the I^{2}C-bus that triggers an I^{2}C 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 I^{2}C can wake up the part independently of whether the I^{2}C interface is configured in master or slave mode.

In deep-sleep mode, the I^{2}C clock is turned off as are all peripheral clocks. However, if the I^{2}C is configured in slave mode and an external master on the I^{2}C-bus provides the clock signal, the I^{2}C interface can create an interrupt asynchronously. This interrupt, if enabled via the POWER_EnterDeepSleep() low power API and in the I^{2}C interface INTENCLR register, can then wake up the core.

33.4.3.1 Wake-up from sleep mode

- Enable the I^{2}C interrupt in the NVIC.
- Enable the I^{2}C 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 94.

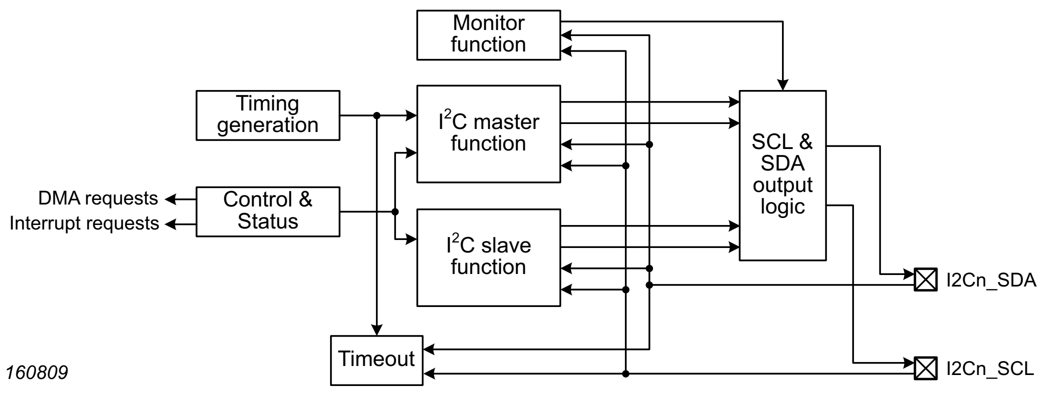


Fig 94. I2C block diagram

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 600. Register overview: I^{2}C register

Name	Access	Offset	Description	Reset value	Section
Shared I^{2}C 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 I^{2}C 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 600. 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 601. 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 601. I^{2}C 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 I^{2}C pins drive and filter, as well as the timing for certain I^{2}C signalling, enabling high-speed mode applies to all functions: master, slave, and monitor.	0
		0	Standard or Fast-modes. The I^{2}C 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 I^{2}C pin,	
		1	High-speed. In addition to Standard-mode, Fast-mode, and Fast-mode Plus, the I^{2}C 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 I^{2}C status register

The STAT register provides status flags and state information about all of the functions of the I^{2}C 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 604 and Table 605.

Table 602. I^{2}C 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 I^{2}C-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 602. 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 604 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 602. I^{2}C 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 I^{2}C-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 I^{2}C interface is in HSCAPABLE mode. See Section 33.7.2.2 “Bus rate support”. When the I^{2}C interface is configured to be HSCAPABLE, HS master codes are detected automatically. Due to the requirements of the HS I^{2}C 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 605 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 I^{2}C 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 I^{2}C bus clock. Deep-sleep mode cannot be entered at this time.		
		1	Not stretching. The slave function is not currently stretching the I^{2}C bus clock. Deep-sleep mode could be entered at this time.		

Table 602. 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 602. 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 603. 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 604. 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 I^{2}C 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 603, if an interrupt is supported there. Reading INTENSET indicates which interrupts are currently enabled.

Table 605. 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 605. 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 606. 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 I^{2}C 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 607. 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 608. 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 603 for detailed descriptions of the interrupt flags.

Table 609. 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 609. 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 603.

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 604, set the MSTDMA bit either with or after the MSTCONTINUE bit. MSTDMA can be cleared at any time.

Table 610. 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 610. 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 I^{2}C 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 I^{2}C 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 I^{2}C 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 I^{2}C master data operations. When this I^{2}C 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 I^{2}C clock pre-divider is described in Table 611.

Table 611. 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 I^{2}C bus specification. I^{2}C 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 611. 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 I^{2}C bus specification. I^{2}C 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 612. 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 I^{2}C slave interface. Only write to this register when the slave is pending (SLVPENDING = 1 in the STAT register,Table 603.

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 605), set the SLVDMA bit either with or after the SLVCONTINUE bit. SLVDMA can be cleared at any time.

Table 613. 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 613. 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 614. 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 I^{2}C slave hardware.

The I^{2}C 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 615. 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 I^{2}C traffic while handling previous I^{2}C data or other operations.	0
		0	Normal operation, matching I^{2}C 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 I^{2}C slave hardware.

Table 616. 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 617. 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 618. 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 618. Monitor data register (MONRXDAT, offset = 0x880) *…continued*

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 I^{2}C module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 619. Module identification register (ID, offset = 0xFFC)

Bit	Symbol	Description	Reset Value
7:0	-	Reserved. Read value is undefined, only zero should be written.	-
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 I^{2}C registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the I^{2}C function.

33.7.2 Bus rates and timing considerations

Due to the nature of the I^{2}C bus, it is generally not possible to guarantee a specific clock rate on the SCL pin. On the I^{2}C-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 I^{2}C traffic (i.e., when it is the only master on the bus, or during arbitration between masters).

In addition, I^{2}C 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 I^{2}C 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 609 and Table 612.

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 621 shows the recommended settings for 400 kHz clock rate.

Table 620. 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 I^{2}C interface is designed to directly control those pad characteristics when configured to be HS capable. The I^{2}C 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 I^{2}C 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 I^{2}C 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, I^{2}C 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 I^{2}C 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 I^{2}C interface can be used to detect a *stuck* bus and potentially do something to alleviate the condition. Two different types of time-outs are supported. Both types apply whenever the I^{2}C 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 I^{2}C clock (SCL). This time-out is asserted when the time between any of these events is longer than the time configured in the TIMEOUT register. This time-out could be useful in monitoring an I^{2}C bus within a system as part of a method to keep the bus running of problems occur.

The second type of I^{2}C time-out is reflected by the SCLTIMEOUT flag in the STAT register. This time-out is asserted when the SCL signal remains low longer than the time configured in the TIMEOUT register. This corresponds to SMBus time-out parameter $T_{TIMEOUT}$. In this situation, a slave could reset its own I^{2}C 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 I^{2}C 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 I^{2}C master sending a second address byte to extend a particular range of standard 7-bit addresses. In the case of the master writing to the slave, the I^{2}C 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 I^{2}C address.

For the master function, the I^{2}C 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 I^{2}C 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 I^{2}C 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 I^{2}C 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. These cases apply when an address matching SLVADR0, qualified by SLVQUAL0, is received.

Table 621. Automatic operation cases

Conditions:			Response:		
AUTONACK bit	**AUTOACK bit**	**Received R/W bit matches AUTOMATCHREAD**	**SLVPENDING interrupt generated?**	**ACK/NACK on I^{2}C 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 621. Automatic operation cases *…continued*

Conditions:			**Response:**		
AUTONACK bit	**AUTOACK bit**	**Received R/W bit matches AUTOMATCHREAD**	**SLVPENDING interrupt generated?**	**ACK/NACK on I^{2}C 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.

UM11424

Chapter 34: LPC55S0x/LPC550x USARTs

Rev. 1.5 — 21 December 2023 **User manual**

34.1 How to read this chapter

USART functions are available on all LPC55S0x/LPC550x 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 4.5.7 "Peripheral reset control 1" or Section 4.5.8 "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 "LPC55S0x/LPC550x 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 FRG associated with the Flexcomm) shall not be:

    – above 25 MHz when the maximum System Frequency chosen by the user is below or equal to 72 MHz.

    – above 100 MHz otherwise (when the maximum System Frequency chosen by the user is above 72 MHz).
- 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 4.5.46 “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.

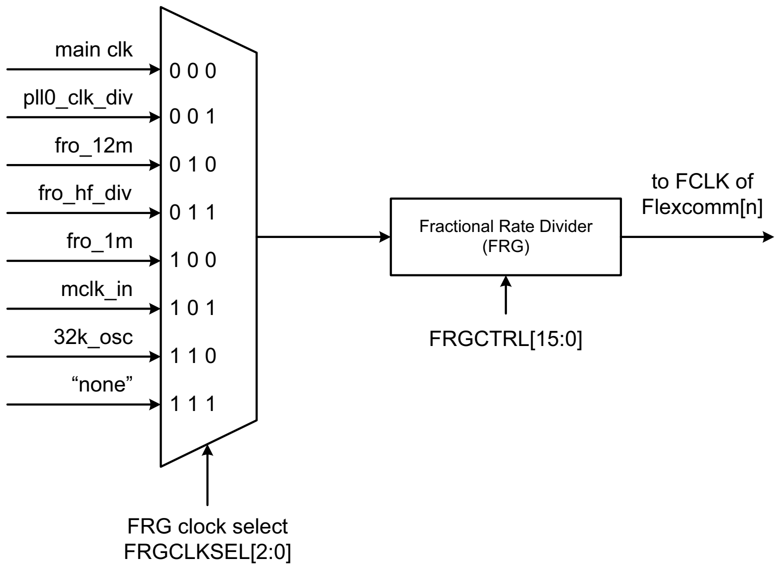


Fig 95. Select clock for FCLK

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 626.
- 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 626. 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, the wake-up time can be customized.

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 Chapter 15 “LPC55S0x/LPC550x I/O Pin Configuration (IOCON)” to assign the USART functions to pins on the device package. Recommended IOCON settings are shown in Table 623.

Table 622. 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 623. 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
11	Reserved	Reserved	SSEL: Generally set to 0

Table 623. Suggested USART pin settings *…continued*

IOCON bit(s)	Type D pin	Type A pin	Type I pin
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.
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.

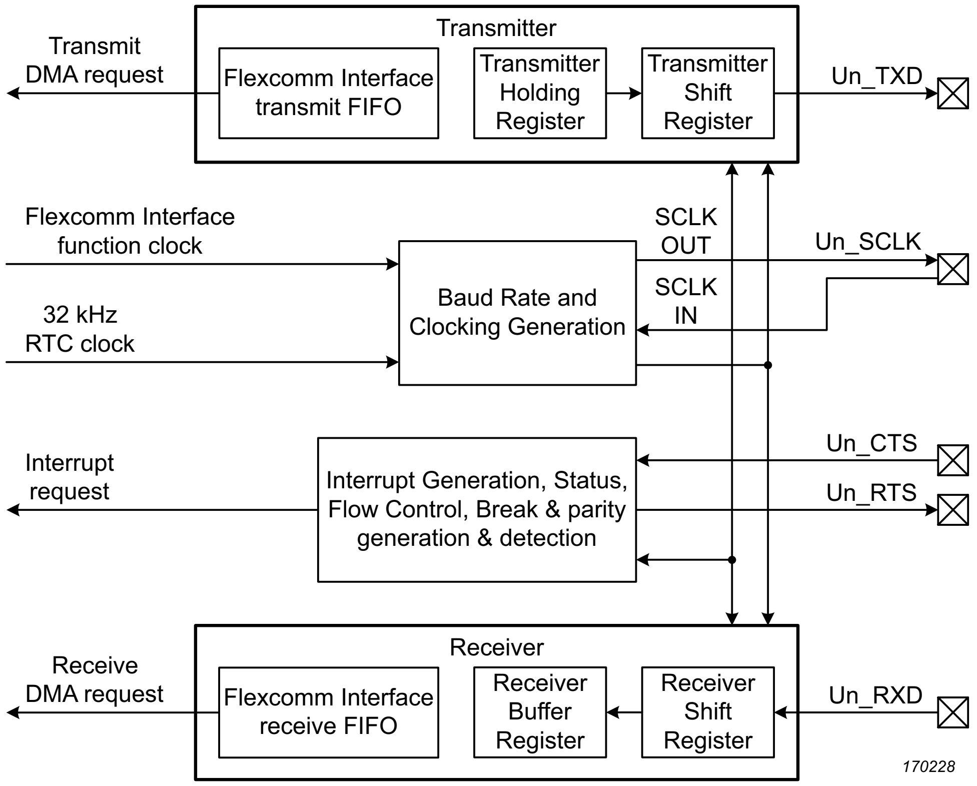

Transmitter
Transmit DMA request
Flexcomm Interface transmit FIFO
Transmitter Holding Register
Transmitter Shift Register
Un_TXD
Flexcomm Interface function clock
SCLK OUT
Un_SCLK
SCLK IN
32 kHz RTC clock
Baud Rate and Clocking Generation
Interrupt request
Interrupt Generation, Status, Flow Control, Break & parity generation & detection
Un_CTS
Un_RTS
Receiver
Receive DMA request
Flexcomm Interface receive FIFO
Receiver Buffer Register
Receiver Shift Register
Un_RXD
170228

Fig 96. 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 624. 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 625. 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. 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.	0x3	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
FIFOINTENSET	R/W1S	0xE10	FIFO interrupt enable set (enable) and read register.	0	34.6.13

Table 625. USART register overview *…continued*

Name	Access	Offset	Description	Reset value	Section
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 626. 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 626. 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 626. 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 627. 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 627. 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 630.

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 628. 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 628. 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 629. 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 628.	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 630. 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 631. 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 628 for detailed descriptions of the interrupt flags.

Table 632. 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 633. 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 634. 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 635. 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 635. 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 636. 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 636. 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 637. 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 637. 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 638. 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 639. 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 640. 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 641. 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 642. 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 642. 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 628.	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 643. 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 644. 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 645. Module identification register (ID - offset 0xFFC)

Bit	Symbol	Description	Reset Value
7:0	-	Reserved. Read value is undefined, only zero should be written.	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 4.5.46 “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 97 shows an overview of RTS and CTS within the USART.

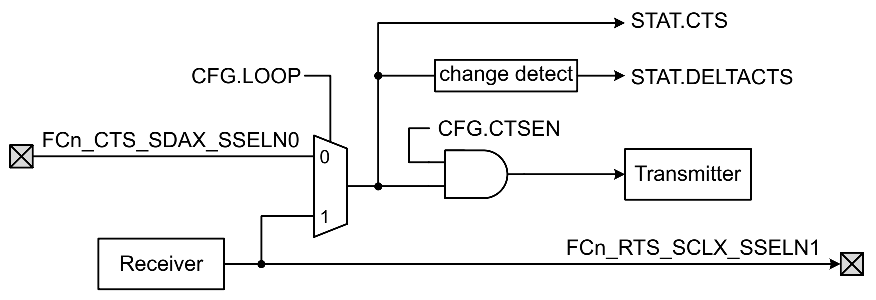


Fig 97. Hardware flow control using RTS and CTS

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.

UM11424

Chapter 35: LPC55S0x/LPC550x Serial Peripheral Interfaces

Rev. 1.5 — 21 December 2023 **User manual**

35.1 How to read this chapter

SPI functions are available on all LPC55S0x/LPC550x 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 be operated with the system DMA controller.
- Data can be transmitted to a slave without the need to read incoming data which 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 in order to only receive data.

The Flexcomm interface function clock frequency (at the output of the FRG associated with the flexcomm) shall not be:

- above 33 MHz when the maximum System Frequency chosen by the user is below or equal to 72 MHz
- above 100 MHz otherwise (when the maximum System Frequency chosen by the user is above 72 MHz)

The High-Speed Flexcomm interface function clock frequency (at the output of the FRG associated with the flexcomm) shall not be:

- above 72 MHz when the maximum System Frequency chosen by the user is below or equal to 72 MHz.
- above 100 MHz otherwise (when the maximum System Frequency chosen by the user is above 72 MHz)

- 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 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 649.
- 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 649. The SCK function must be connected to a pin that is 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 647.

Table 646. 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.

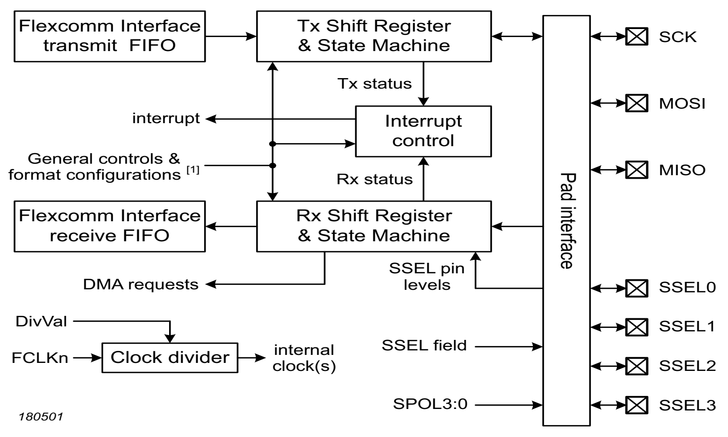
Table 647. Suggested SPI pin settings

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 647. 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 98. 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 648. 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

Table 648. SPI register overview *…continued*

Name	Access	Offset	Description	Reset value	Section
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	35.6.18
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 651 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 "LPC55S0x/LPC550x 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 651 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 649. 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.	
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 650. 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 651. 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 651 for details of the interrupts.

Table 652. 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.	

Table 652. SPI interrupt enable read and set register (INTENSET, offset = 0x40C) *…continued*

Bit	Symbol	Value	Description	Reset value
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 653. 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 654. 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 651 for detailed descriptions of the interrupt flags.

Table 655. 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 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. 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.	-	
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.	-	-

35.6.10 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	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

Table 657. FIFO status register (FIFOSTAT - offset = 0xE04) *…continued*

Bit	Symbol	Description	Reset value	Access
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 658. 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.	
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

Table 658. FIFO trigger 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. ... 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 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	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.	
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 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.	-

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 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.	-

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 662. 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 662. 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 reading unneeded received data. No receiver flags are generated.	

Table 662. 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 having 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 be 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 663. 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 664. 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 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 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 666. Module identification register (ID, offset = 0xFFC)

Bit	Symbol	Description	Reset value
7:0	-	Reserved. Read value is undefined, only zero should be written.	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 667 and shown in Figure 99. CPOL and CPHA are configured by bits in the CFG register. See Section 35.6.2 “SPI configuration register”.

Table 667. 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

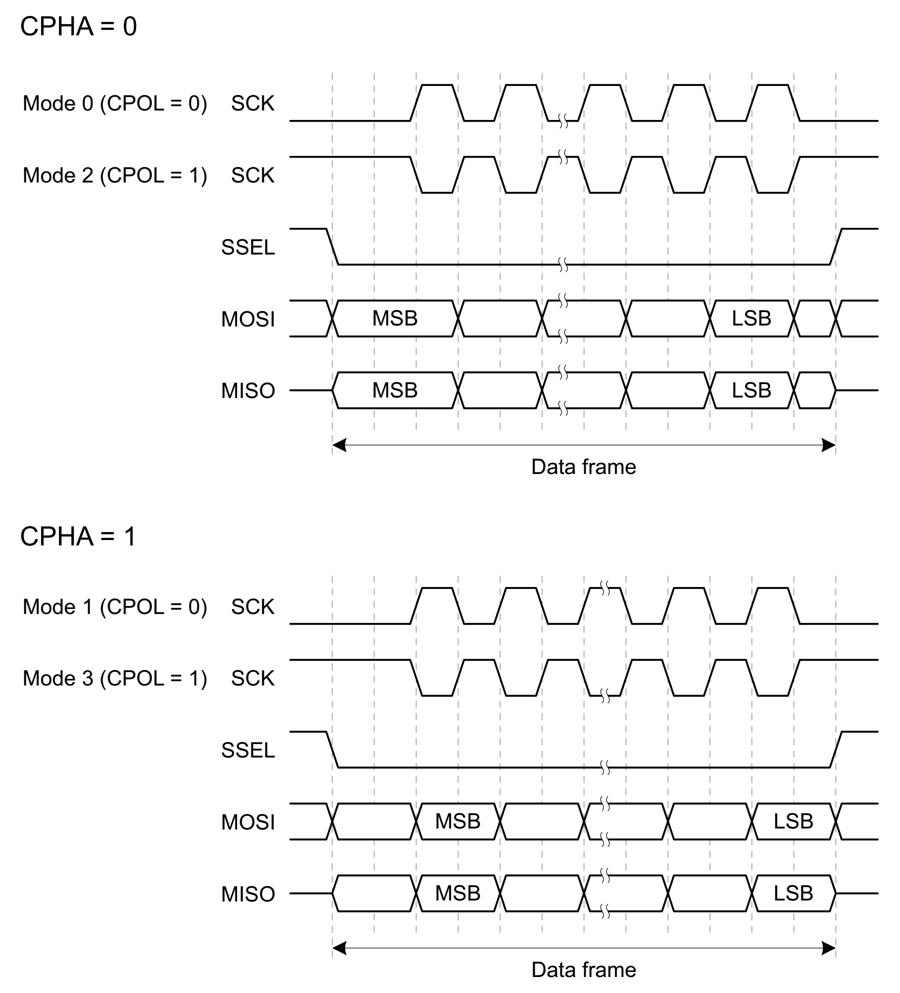


Fig 99. Basic SPI operating mode

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 100. 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.

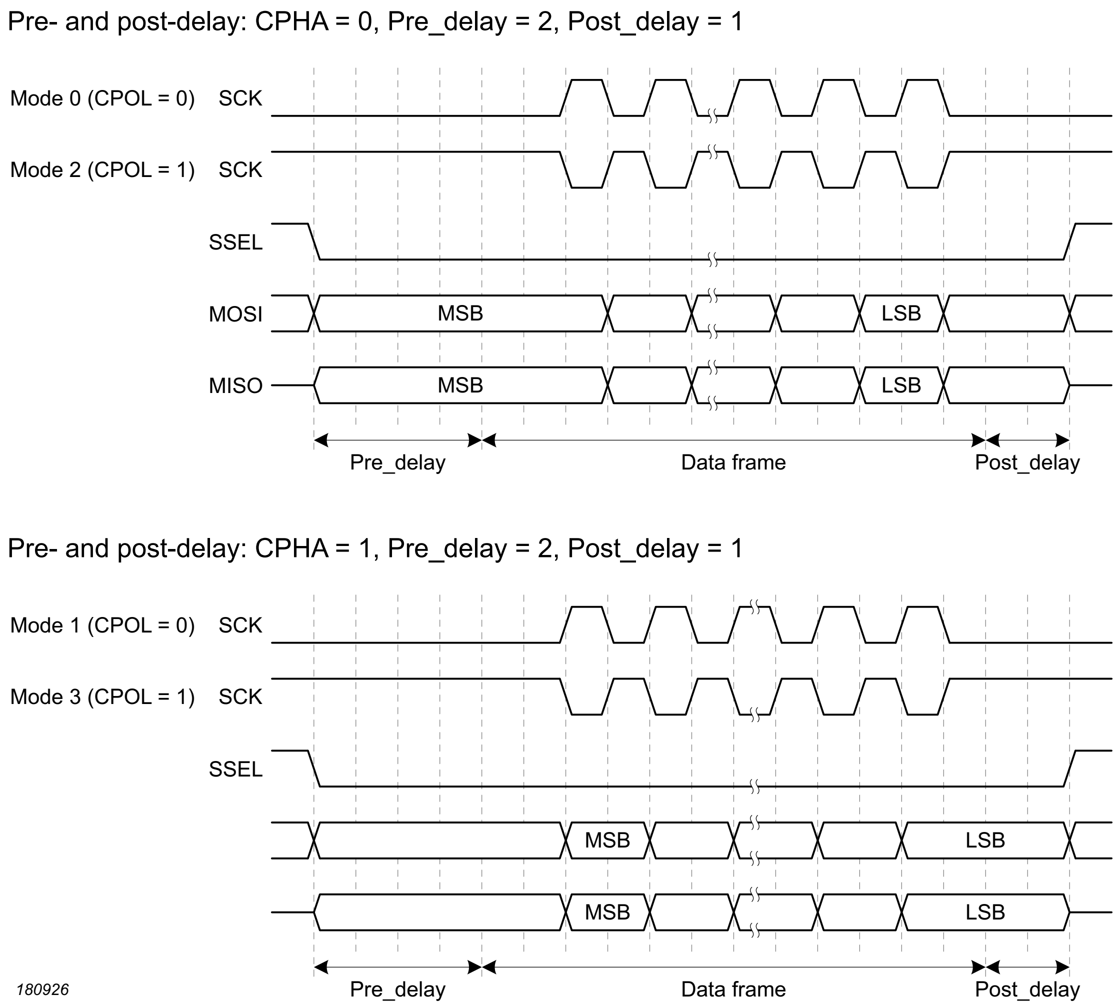


Fig 100. Pre_delay and Post_delay

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 101. 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.

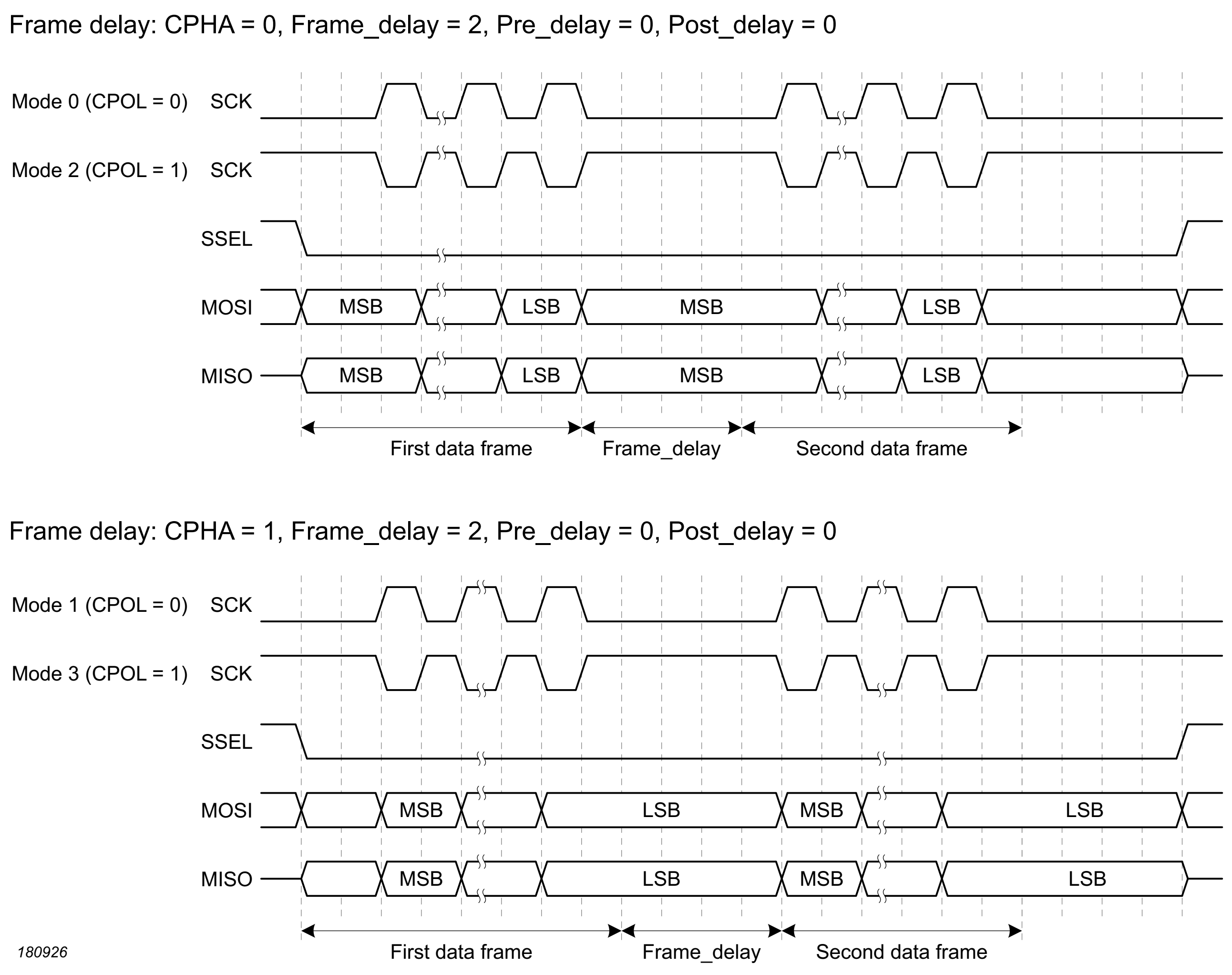


Fig 101. Frame_delay

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 102.

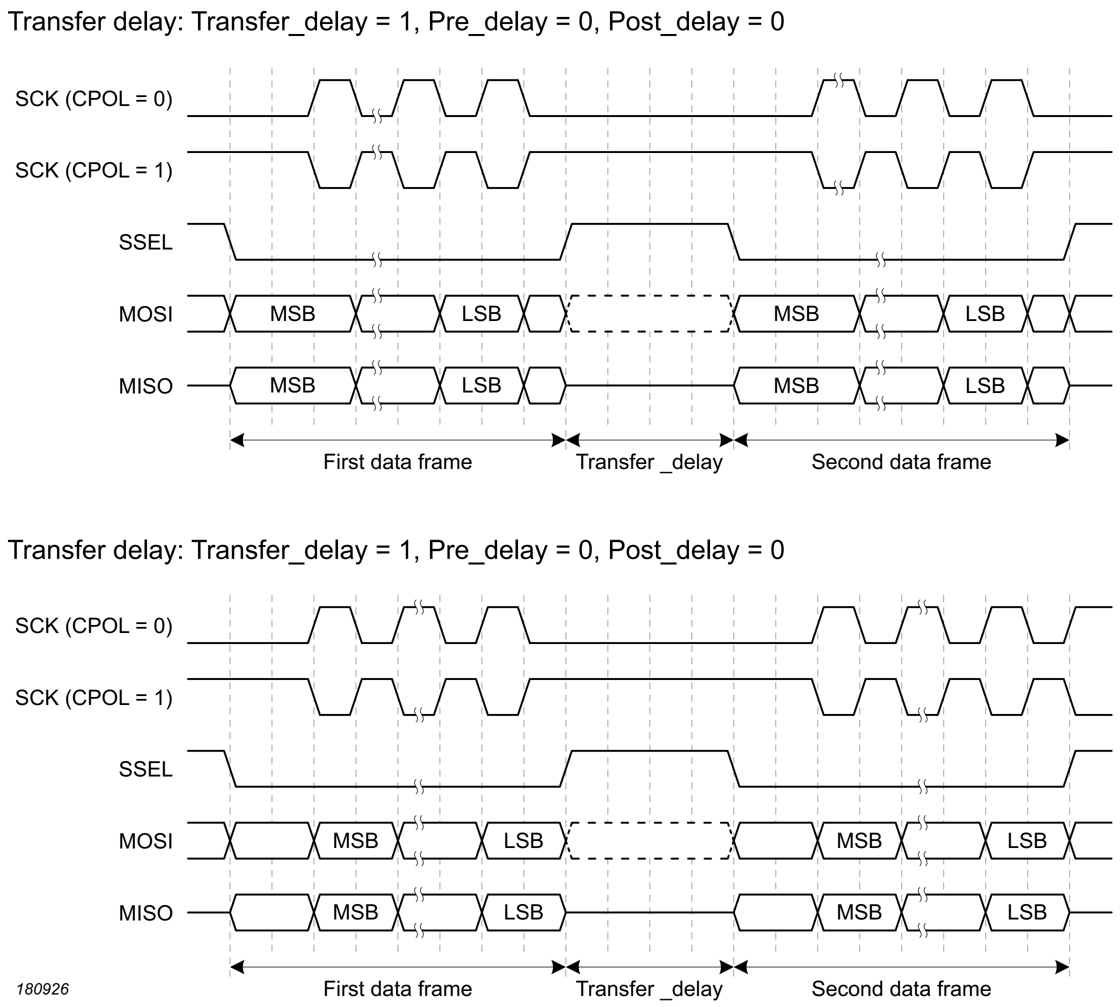

Transfer delay: Transfer_delay = 1, Pre_delay = 0, Post_delay = 0
SCK (CPOL = 0)
SCK (CPOL = 1)
SSEL
MOSI
MISO
MSB
LSB
First data frame
Transfer _delay
Second data frame
Transfer delay: Transfer_delay = 1, Pre_delay = 0, Post_delay = 0
180926

Fig 102. 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.

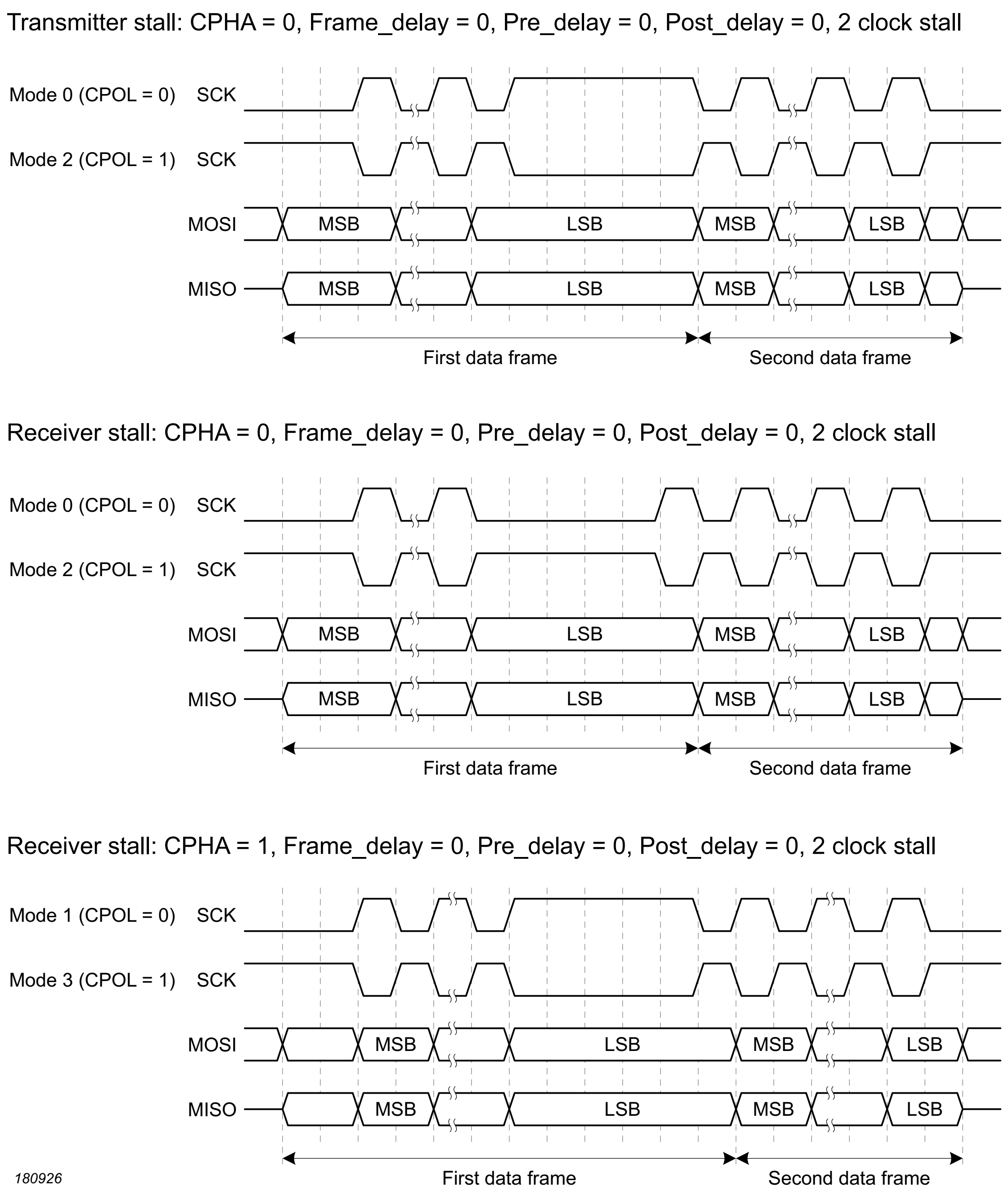

Transmitter stall: CPHA = 0, Frame_delay = 0, Pre_delay = 0, Post_delay = 0, 2 clock stall
Mode 0 (CPOL = 0) SCK
Mode 2 (CPOL = 1) SCK
MOSI
MISO
MSB
LSB
First data frame
Second data frame
Receiver stall: CPHA = 0, Frame_delay = 0, Pre_delay = 0, Post_delay = 0, 2 clock stall
Receiver stall: CPHA = 1, Frame_delay = 0, Pre_delay = 0, Post_delay = 0, 2 clock stall
Mode 1 (CPOL = 0) SCK
Mode 3 (CPOL = 1) SCK
180926

Fig 103. Examples of data stalls

UM11424

Chapter 36: LPC55S0x/LPC550x Sys_ctrl

Rev. 1.5 — 21 December 2023 User manual

36.1 How to read this chapter

The sys_ctrl contains I^{2}S signal sharing. This feature is available on all LPC55S0x/LPC550x devices. A gray to binary decoder is present in sys_ctrl.

36.2 Features

- I^{2}S signal sharing: allows multiple Flexcomm Interface I^{2}S interfaces to share some combination of I^{2}S clock, WS, and DATA without external board wiring.
- Gray to binary decoder: allows decoding gray value coming from OS Event Timer.

36.3 Basic configuration

36.3.1 I^{2}S signal sharing

Configure I^{2}S 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 I^{2}S 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 I^{2}S 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 I^{2}S master that is providing SCK and WS signals for shared usage should also be configured to use the shared signal. For example, if the 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

I^{2}S signal sharing does not directly use pins, but offers additional internal routing of existing I^{2}S pin functions.

36.5 Register description

Table 668. 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
CODE_GRAY_LSB	RW	0x180	CODE_GRAY LSB input Register.	0x0	36.5.4
CODE_GRAY_MSB	RW	0x184	CODE_GRAY MSB input Register.	0x0	36.5.5
CODE_BIN_LSB	RO	0x188	CODE_BIN LSB output Register.	0x0	36.5.6
CODE_BIN_MSB	RO	0x18C	CODE_BIN MSB output Register.	0x0	36.5.7

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 669. 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 670 for details on how shared signals are connected and selected.

Table 670. Shared signal control select registers for each Flexcomm (FC0CTRLSEL to FC7CTRLSEL, offset 0x040 to 0x05C)

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.	

Table 670. Shared signal control select registers for each Flexcomm (FC0CTRLSEL to FC7CTRLSEL, offset 0x040 to 0x05C) *…continued*

Bit	Symbol	Value	Description	Reset value
7:2	-		Reserved.	undefined
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.

Table 671. 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 671. 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 CODE GRAY for LSB input

Serves at the input register for CODE GRAY LSB.

Table 672. CODE_GRAY LSB input Register (CODE_GRAY_LSB, offset 0x180)

Bit	Symbol	Value	Description	Reset value
31:0	CODE_GRAY_LSB	RW	Gray code (42bits) to be converted back to binary.	0x0

36.5.5 CODE GRAY for MSB input

Serves as the input register for CODE GRAY MSB.

Table 673. CODE_GRAY MSB input Register (CODE_GRAY_MSB, offset 0x184)

Bit	Symbol	Value	Description	Reset value
9:0	CODE_GRAY_MSB	RW	Gray code (42bits) to be converted back to binary.	0x0
31:10	-	-	Reserved.	Undefined

36.5.6 CODE BIN LSB input

Serves as input register for CODE BIN LSB.

Table 674. CODE_BIN LSB output Register (CODE_BIN_LSB, offset 0x188)

Bit	Symbol	Value	Description	Reset value
9:0	CODE_BIN_LSB		Binary converted code (42bits).	0x0
31:10	-	-	Reserved.	Undefined

36.5.7 CODE BIN MSB input

Serves as the input register for CODE BIN MSB.

Table 675. CODE_BIN MSB output Register (CODE_BIN_MSB, offset 0x18C)

Bit	Symbol	Value	Description	Reset value
9:0	CODE_BIN_MSB	RW	Binary converted code (42bits).	0x0
31:10	-	-	Reserved.	Undefined

36.6 Functional description

36.6.1 I^{2}S signal sharing

The I^{2}S signal sharing features are available on all LPC55S0x/LPC550x devices.

It is sometimes desirable to use multiple I^{2}S functions together in a single TDM stream without sacrificing more pins than are needed. I^{2}S 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 I^{2}S interface that includes four channel pairs.

Signal sharing allows more than one on-chip I^{2}S interface to be connected to clock, WS, and input data on the same pins without external board wiring. Multiple I^{2}S functions contributing output data to a single data line must still be accomplished with an external connection.

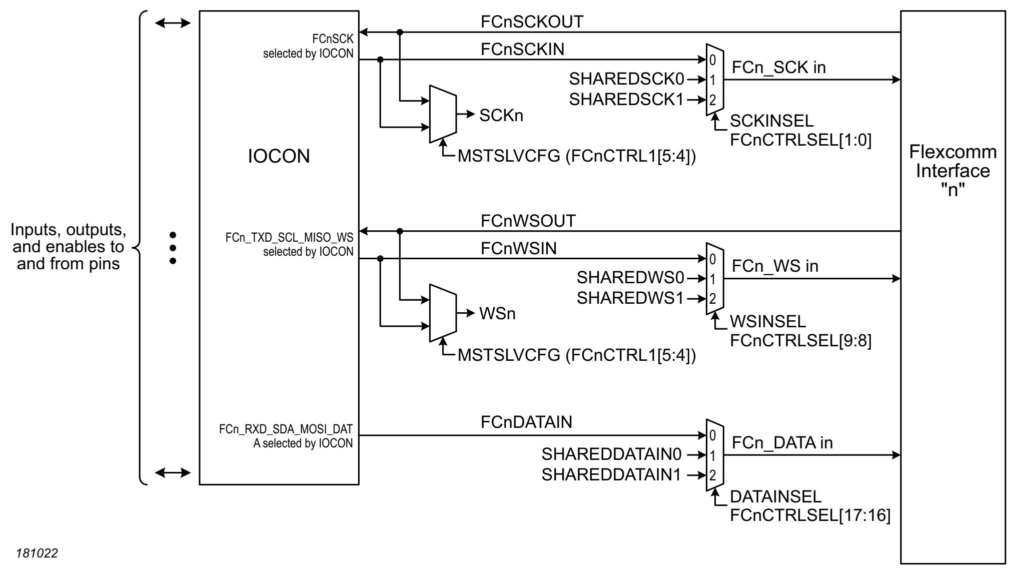
In general, each Flexcomm Interface configured for I^{2}S can choose:

- 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:

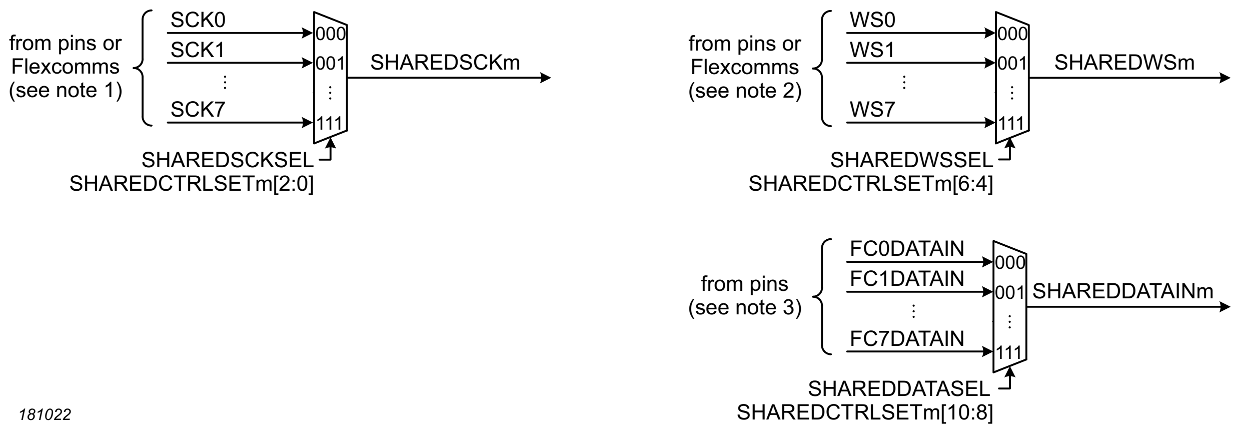
- Its own SCK (in or out, depending on whether it is a master or slave).
- Its own WS (in or out, depending on whether it is a master or slave).
- Its own DATA input.

Representative logic for the connection possibilities are shown in Figure 104 and Figure 105.



Note: these connection options are replicated for each Flexcomm "n", for all Flexcomm Interfaces with I2S support.

Fig 104. Shared signal connections for each Flexcomm Interface


from pins or Flexcomms (see note 1)
SCK0
SCK1
SCK7
000
001
111
SHAREDSCKm
SHAREDSCKSEL SHAREDCTRLSETm[2:0]
from pins or Flexcomms (see note 2)
WS0
WS1
WS7
SHAREDWSm
SHAREDWSSEL SHAREDCTRLSETm[6:4]
from pins (see note 3)
FC0DATAIN
FC1DATAIN
FC7DATAIN
SHAREDDATAINm
SHAREDDATASEL SHAREDCTRLSETm[10:8]
181022

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 105. Shared signal source selection and control

36.6.1.1 Examples

Figure 106 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

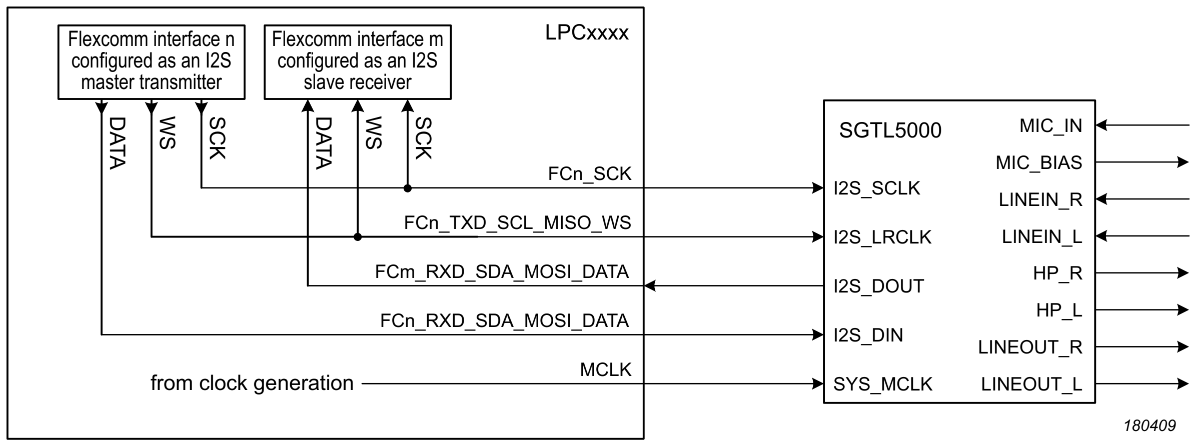
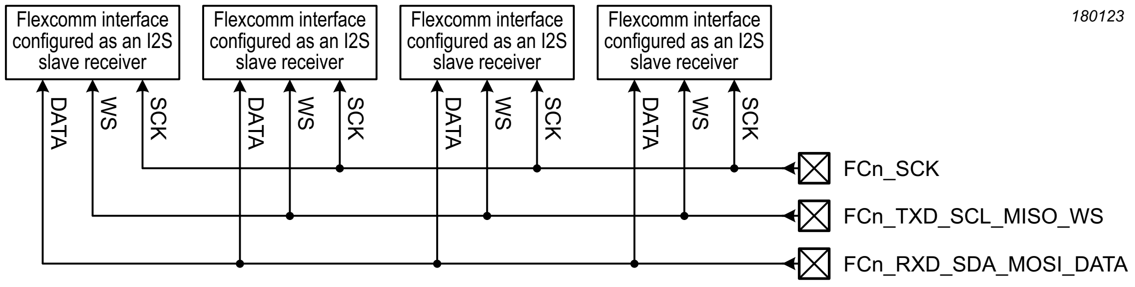


Fig 106. Example connection to an I^{2}S bidirectional codec

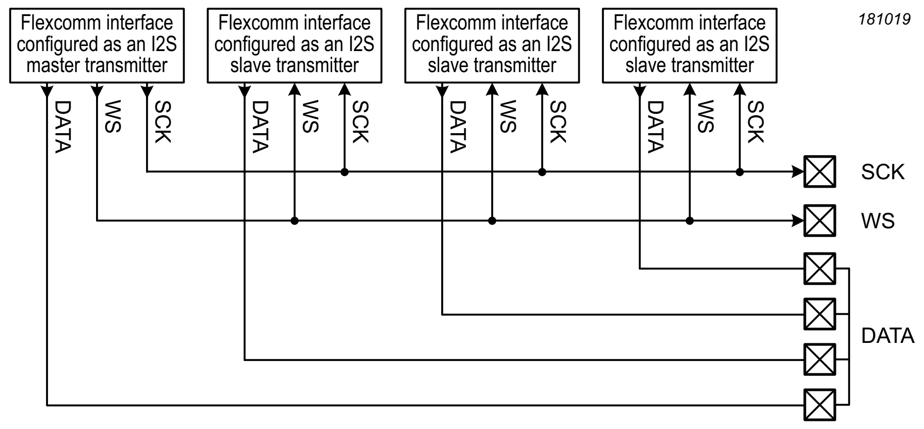
Figure 107 shows a generic case of multiple slaves and/or receivers sharing SCK and WS, and/or DATA. This scenario includes received data sharing (e.g., different I^{2}S interfaces receiving data from different slots in a TDM stream).


Flexcomm interface configured as an I2S slave receiver
Flexcomm interface configured as an I2S slave receiver
Flexcomm interface configured as an I2S slave receiver
Flexcomm interface configured as an I2S slave receiver
DATA
WS
SCK
FCn_SCK
FCn_TXD_SCL_MISO_WS
FCn_RXD_SDA_MOSI_DATA
180123

All I2S interfaces are slave receivers sharing SCK, WS, and DATA.

Fig 107. I^{2}S signal sharing example, multiple slave receivers

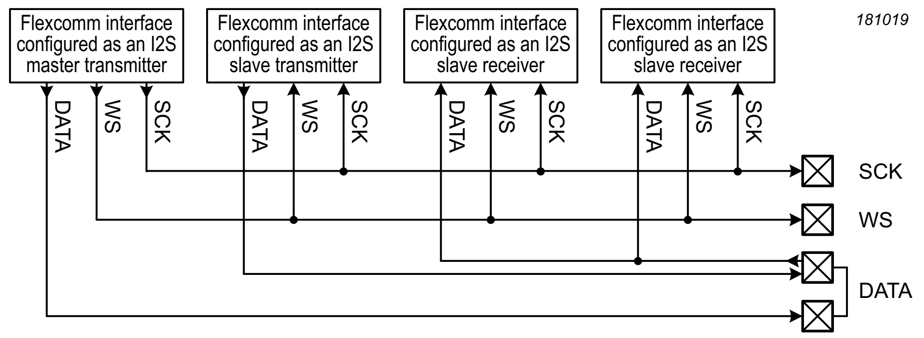
Figure 108 shows master to slave operation where one I^{2}S interface is a master going off chip, and other on-chip I^{2}S interfaces are slaved to it. Data could be either transmitted or received. Multiple I^{2}S interfaces supply data to a single stream by wiring multiple pins together.


181019
Flexcomm interface configured as an I2S master transmitter
Flexcomm interface configured as an I2S slave transmitter
Flexcomm interface configured as an I2S slave transmitter
Flexcomm interface configured as an I2S slave transmitter
DATA
WS
SCK
SCK
WS
DATA

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 108. I^{2}S signal sharing example, one master and multiple slave transmitters

Figure 109 shows data with one I^{2}S interface transmitting onto a shared DATA line while at least one other I^{2}S 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 I^{2}S interfaces transmitting and two receiving, but it could be any combination.


181019
Flexcomm interface configured as an I2S master transmitter
Flexcomm interface configured as an I2S slave transmitter
Flexcomm interface configured as an I2S slave receiver
Flexcomm interface configured as an I2S slave receiver
DATA
WS
SCK
SCK
WS
DATA

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 109. I^{2}S signal sharing example, one master with mixed transmitters and receivers

UM11424

Chapter 37: LPC55S0x/LPC550x I^{2}S interface

Rev. 1.5 — 21 December 2023 **User manual**

37.1 How to read this chapter

I^{2}S functionality is available on all LPC55S0x/LPC550x devices. I^{2}S is a function that is implemented in selected Flexcomm Interfaces. The I^{2}S function will be enabled in all Flexcomm Interfaces, each with 1x I^{2}S channel pair (each I^{2}S channel pair can handle transmitted or received stereo data). Flexcomm Interface 6 and 7 have four I2S channel pairs and other flexcomms have only one channel pair.

The I^{2}S 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 the clocking diagram for I^{2}S clocking.

I^{2}S pin sharing logic to support full-duplex I^{2}S operation from common pins is described in Chapter 36 "LPC55S0x/LPC550x Sys_ctrl".

37.2 Features

The I^{2}S bus provides a standard communication interface for streaming data transfer applications such as digital audio or data collection. The I^{2}S 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 I^{2}S 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 I^{2}S 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 I^{2}S block in the system level clocking scheme.

- A Flexcomm Interface may implement one or more I2S channel pairs. The first channel pair can be either a master or a slave, and the rest of the channel pairs are always slaves. All channel pairs are configured together for either transmit or receive and other shared attributes. The number of channel pairs is defined for Flexcomm Interface 6 and 7.
- 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).
- All channel pairs within one Flexcomm Interface share a single bit clock (SCK) and word select/frame trigger (WS), and data line (SDA).
- Data for all I^{2}S 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 I^{2}S peripheral is accomplished as follows:

1. Peripheral clock: Make sure that the related Flexcomm Interface is enabled in the AHBCLKCTRL1 register. See Section 4.5.17 “AHB clock control 1”.
2. Flexcomm Interface clock: Select a clock source for the related Flexcomm Interface. Options are shown in Figure 3. Also see Section 4.5.39 “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 25 MHz when the maximum System Frequency chosen by the user is below or equal to 72 MHz.

   – above 100 MHz otherwise (when the maximum System Frequency chosen by the user is above 72 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 4.5.16 “AHB clock control 0”. Select I^{2}S pins and pin modes through the relevant IOCON registers, see Chapter 15 “LPC55S0x/LPC550x I/O Pin Configuration (IOCON)”.
6. I^{2}S rate: For master operation, the I^{2}S rate is determined by the clock selected in step 2, optionally modified using the DIV register, see Table 682. Slave functions typically use the incoming I^{2}S clock directly.
7. Interrupts: To enable I^{2}S 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: I^{2}S 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 110 for the overall architecture of an example I2S subsystem.

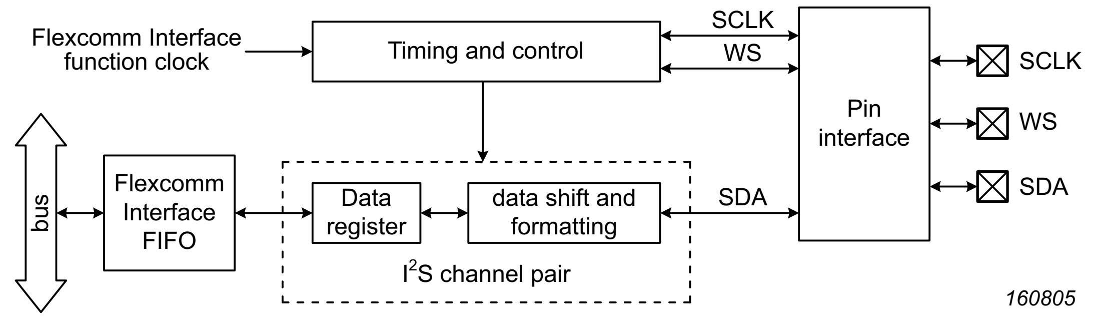


Fig 110. I2S block diagram

37.5 Terminology

Table 676. 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 micro controller, 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 I^{2}S function is outputting SCK and/or WS, it uses a return signal from the related pin to adjust internal timing. In order for the I^{2}S to operate, the signals must be connected to a device pin, via IOCON selection.

Table 677. I^{2}S pin description

Pin	Type	Name used in pin configuration chapter	Description
SCK	I/O	FCn_SCK	Serial clock for I^{2}Sn. 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 I^{2}S 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 I^{2}S 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 I^{2}S 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 I^{2}S to work properly.
SDA	I/O	FCn_RXD_SDA_MOSI	Serial data for a single data stream used by one or more I^{2}S channel pairs of I^{2}Sn. 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 I^{2}S block. If MCLK is supported as an input to the device, it can be routed to the I^{2}S 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 I^{2}S block and used to operate its functions.

37.7 Register description

The registers shown in Table 678 apply if the I^{2}S function is selected in a Flexcomm Interface that supports I^{2}S. 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:

I^{2}S0 base address: 4008_6000h

I^{2}S1 base address: 4008_7000h

I^{2}S2 base address: 4008_8000h

I^{2}S3 base address: 4008_9000h

I^{2}S4 base address: 4008_A000h

I^{2}S5 base address: 4009_6000h

I^{2}S6 base address: 4009_7000h

I^{2}S7 base address: 4009_8000h

Table 678. Register overview for the I^{2}S 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 I^{2}S 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 I^{2}S 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 678. Register overview for the I^{2}S 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 I^{2}S is configured for 2x 24-bit data and not using DMA.	-	37.7.16
FIFOSIZE	R	0xE48	FIFO size.	0x8	
Registers for secondary channel pairs:					
P1CFG1	R/W	0xC20	Configuration register 1 for channel pair 1.	0	37.7.18
P1CFG2	R/W	0xC24	Configuration register 2 for channel pair 1.	0	37.7.18
P1STAT	RO/W1C	0xC28	Status register for channel pair 1.	0	37.7.19
P2CFG1	R/W	0xC40	Configuration register 1 for channel pair 2.	0	37.7.18
P2CFG2	R/W	0xC44	Configuration register 2 for channel pair 2.	0	37.7.18
P2STAT	RO/W1C	0xC48	Status register for channel pair 2.	0	37.7.19
P3CFG1	R/W	0xC60	Configuration register 1 for channel pair 3.	0	37.7.18
P3CFG2	R/W	0xC64	Configuration register 2 for channel pair 3.	0	37.7.18
P3STAT	RO/W1C	0x68	Status register for channel pair 3.	0	37.7.19
ID register:					
ID	RO	0xFFC	I^{2}S module identification. This value appears in the shared Flexcomm Interface peripheral ID register when I^{2}S is the selected function.	0xE090	37.7.21

[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 I^{2}S channel pairs within one Flexcomm Interface. A few settings apply only to the primary channel pair, as noted.

Table 679. Configuration register 1 (CFG1, offset = 0xC00)

Bit	Symbol	Value	Description	Reset value
0	MAINENABLE		Main enable for I^{2}S function in this Flexcomm Interface.	0
		0	All I^{2}S 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 I^{2}S 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 I^{2}S 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 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 I^{2}S 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 I^{2}S 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 I^{2}S channel pair in this Flexcomm Interface. 01 = there are two I^{2}S channel pairs in this Flexcomm Interface. 10 = there are three I^{2}S channel pairs in this Flexcomm Interface. 11 = there are four I^{2}S 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 679. Configuration register 1 (CFG1, offset = 0xC00) *…continued*

Bit	Symbol	Value	Description	Reset value
7:6	MODE		Selects the basic I^{2}S operating mode. Other configurations modify this to obtain all supported cases. See Section 37.8.2 “Formats and modes” for examples.	0
		0x0	I^{2}S 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 I^{2}S 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 I^{2}S 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 I^{2}S 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 I^{2}S channel pair. Other channel pairs may select this mode independently in their separate CFG1 registers.	0
		0	I^{2}S data for this channel pair is treated as left and right channels.	
		1	I^{2}S 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 679. 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 680. Configuration register 2 (CFG2, offset = 0xC04)

Bit	Symbol	Description	Reset value
10:0	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 ... 0x7FF = frame is 2048 bits in total length **Remark:** If FRAMELEN 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
15:11	-	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 I^{2}S 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 681.

Table 681. 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 681. 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 I^{2}S 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 I^{2}S 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	DATAPAUSED		Data paused status flag. Applies to all I^{2}S 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 I^{2}S function, which is the case when the MSTSLVCFG field in the CFG1 register = 0 or 2.

Table 682. Clock divider register (DIV, offset = 0xC1C)

Bit	Symbol	Description	Reset value
11:0	DIV	This field controls how this I^{2}S 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 I^{2}S 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 I^{2}S data direction must be enabled because the FIFO is the only means for accessing I^{2}S data.

Table 683. 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 I^{2}S empty 0.Determines the value sent by the I^{2}S in transmit mode if the TX FIFO becomes empty. This value is sent repeatedly until the I^{2}S is paused, the error is cleared, new data is provided, and the I^{2}S 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 I^{2}S FIFO entries are handled as all 24-bit values.		
		1	48-bit I^{2}S 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 I^{2}S. 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 I^{2}S 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 I^{2}S 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 683. 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 I^{2}S 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 684. 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 I^{2}S 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 685. 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 I^{2}S 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 686. 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 687. 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 I^{2}S channels in a single Flexcomm Interface move data in the same direction, only TX and RX related flags and controls are meaningful at any particular time.

Table 688. 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 I^{2}S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 689. 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 I^{2}S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 690. 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 I^{2}S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 691. 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 I^{2}S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 692. 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 693. 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 694. 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 695. 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 Configuration register 1 for channel pairs 1, 2, and 3

The P1CFG1, P2CFG1, and P3CFG1 registers contain mode settings that apply to channel pairs other than the first pair, within the same Flexcomm Interface.

Table 696. Configuration register 1 for channel pairs 1, 2, and 3 (P1CFG1 - offset 0xC20; P2CFG1 - offset 0xC40; P3CFG1 - offset 0xC60) bit description

Bit	Symbol	Value	Description	Reset value
0	PAIRENABLE		Enable for this channel pair.	0
		0	This I^{2}S channel pair is disabled.	
		1	This I^{2}S channel pair is enabled. Other channel pairs in this Flexcomm Interface may be enabled in their individual PAIRNABLE bits.	
9:1	-	-	Reserved. Read value is undefined, only zero should be written.	-
10	ONECHANNEL		Single channel mode. Applies to both transmit and receive. This configuration bit applies only to this I^{2}S channel pair. Other channel pairs may select this mode independently in their separate CFG1 registers.	0
		0	I^{2}S data for this channel pair is treated as left and right channels.	
		1	I^{2}S data for this channel pair is treated as a single channel, functionally the left channel for this pair.	
31:11	-	-	Reserved. Read value is undefined, only zero should be written.	-

37.7.19 Configuration register 2 for channel pairs 1, 2, and 3

These registers contain the frame position for channel pairs beyond the main pair.

Table 697. Configuration register 2 channel pairs 1, 2, 3 (P1CFG2 - offset 0xC24; P2CFG2 - offset 0xC44; P1CFG2 - offset 0xC64)

Bit	Symbol	Description	Reset value
15:0	-	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. See details in the description of POSITION for the primary channel pair.	0
31:25	-	Reserved. Read value is undefined, only zero should be written.	-

37.7.20 Status registers for channel pairs 1, 2, and 3

These read-only registers provide status flags for additional channel pairs beyond the primary channel pair.

Table 698. Status registers for channel pairs 1, 2, and 3 (P1STAT - offset 0xC28; P2STAT - offset 0xC48; P3STAT - offset 0xC68) bit description

Bit	Symbol	Value	Description	Reset value
0	BUSY		Busy status for this channel pair.	0
		0	The transmitter/receiver for this channel pair is currently idle.	
		1	The transmitter/receiver for this channel pair is currently processing data.	
1	SLVFRMERR		Save Frame Error flag. This flag is shared by the STAT and PnSTAT registers. Refer to the STAT register description for details.	0
2	LR		Left/Right indication. This flag is shared by the STAT and PnSTAT registers. Refer to the STAT register description for details.	0
3	DATAPAUSED		Data Paused status flag. This flag is shared by the STAT and PnSTAT registers.	0
		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.	-

37.7.21 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 699. Module identification register (ID, offset = 0xFFC)

Bit	Symbol	Description	Reset value
7:0	-	Reserved. Read value is undefined, only zero should be written.	-
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 I^{2}S registers contained in the Flexcomm Interface support only word writes. Byte and half-word writes are not supported in conjunction with the I^{2}S 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 I^{2}S. 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 I^{2}S 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:

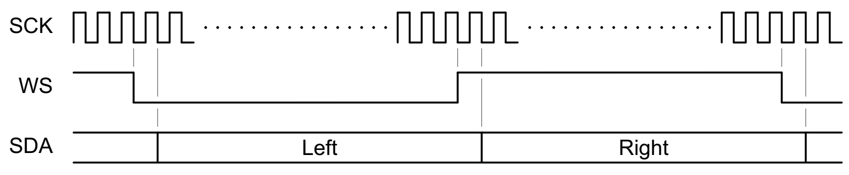
  1) Determines the size of data transfers between the FIFO and the I^{2}S serializer/deserializer.

  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.

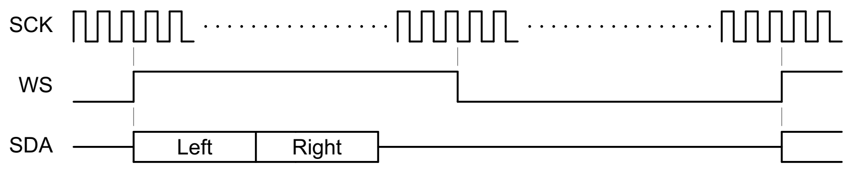
37.8.2.2 Example frame configurations

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.



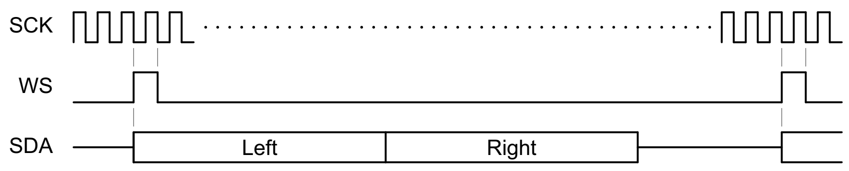
MODE = 0; POSITION = 0; SCK_POL = 0; WS_POL = 0; MONO = 0

Fig 111. Classic I^{2}S mode



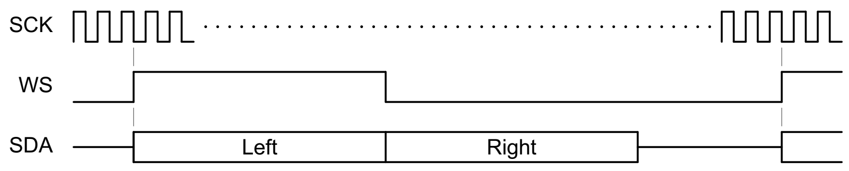
MODE = 1; POSITION = 0; SCK_POL = 0; WS_POL = 1; MONO = 0

Fig 112. DSP mode with 50% WS



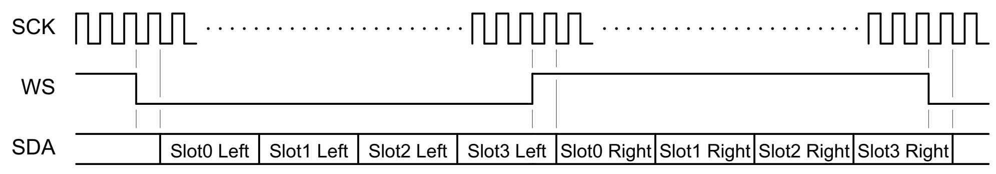
MODE = 2; POSITION = 0; SCK_POL = 0; WS_POL = 1; MONO = 0

Fig 113. DSP mode with 1 SCK pulsed WS



MODE = 3; POSITION = 0; SCK_POL = 0; WS_POL = 1; MONO = 0

Fig 114. DSP mode with 1 slot pulsed WS

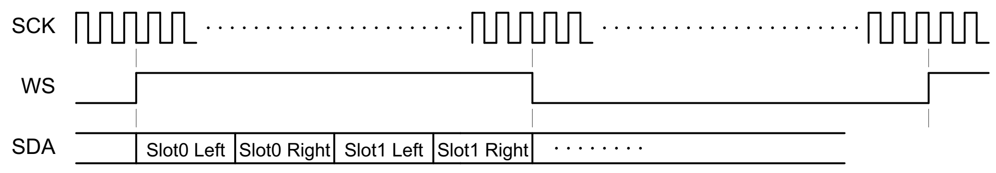


MODE = 0; SCK_POL = 0; WS_POL = 0; MONO = 0

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 I^{2}S channel pair. This example shows 4 data slots.

Fig 115. TDM in classic I^{2}S mode

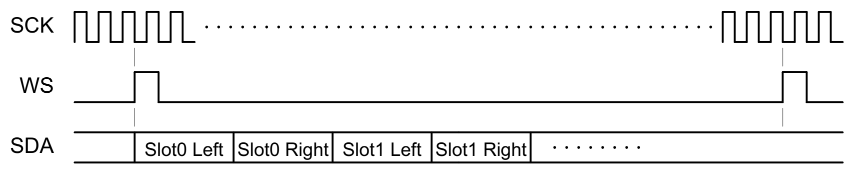


MODE = 1; SCK_POL = 0; WS_POL = 1; MONO = 0

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 116. TDM and DSP modes with 50% WS

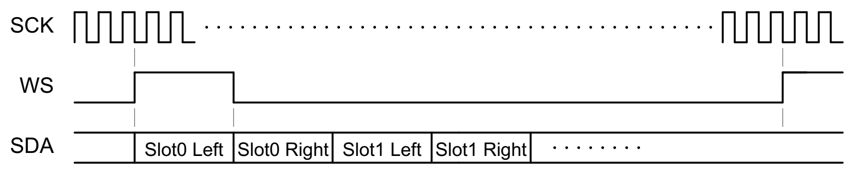


MODE = 2; SCK_POL = 0; WS_POL = 1; MONO = 0

POSITION = bit position of the first used data bit for each slot.

One Left/Right slot would be used by one channel pair.

Fig 117. TDM and DSP modes with 1 SCK pulsed WS

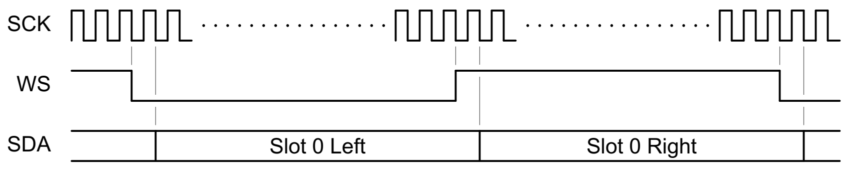


MODE = 3; SCK_POL = 0; WS_POL = 1; MONO = 0

POSITION = bit position of the first used data bit for each slot.

One Left/Right slot would be used by one channel pair.

Fig 118. TDM and DSP modes with 1 slot pulsed WS

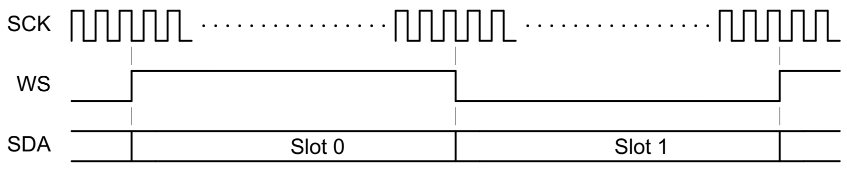


MODE = 0; SCK_POL = 0; WS_POL = 0; MONO = 1.

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 119. I²S mode, mono

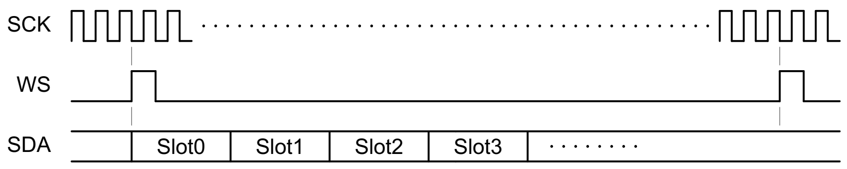


MODE = 1; SCK_POL = 0; WS_POL = 1; MONO = 1

POSITiON = bit position of the first used data bit for each slot.

One slot would be used by one I²S.

Fig 120. 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 121. TDM and DSP modes, mono, with WS pulsed for one SCK time

37.8.2.3 I^{2}S signal polarities

Figure 122 shows examples of SCK and WS polarities and how they relate to data positions.

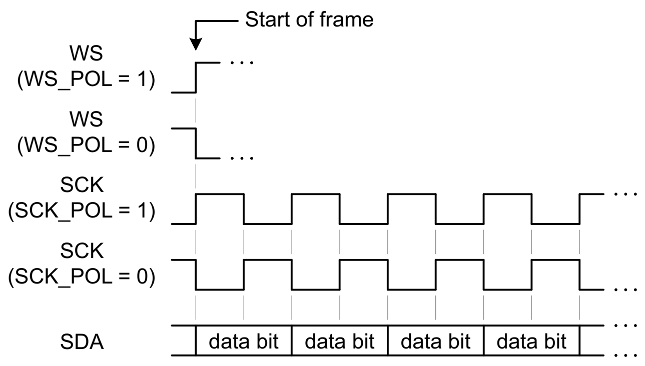


Fig 122. Data at the start of a frame, shown with both SCK and WS polarities

37.8.3 Data rates

37.8.3.1 Rate support

The actual I^{2}S 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 I^{2}S 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 I^{2}S 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 I^{2}S function divider can be used to produce the desired frequency.

Example 1

This I^{2}S 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 I^{2}S 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 I^{2}S 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 I^{2}S 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 I^{2}S clocking has been configured, that channel has been configured, DMA has been configured, and the I^{2}S bus is running. DMA operation is similar to any other serial peripheral.

DMA related configurations in the Flexcomm Interface I^{2}S 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 I^{2}S 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.

UM11424

Chapter 38: LPC55S0x/LPC550x Programmable Logical Unit

Rev. 1.5 — 21 December 2023 **User manual**

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 700. 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 123 shows the PLU block diagram with the following configuration: 26 LUT elements; six primary inputs.

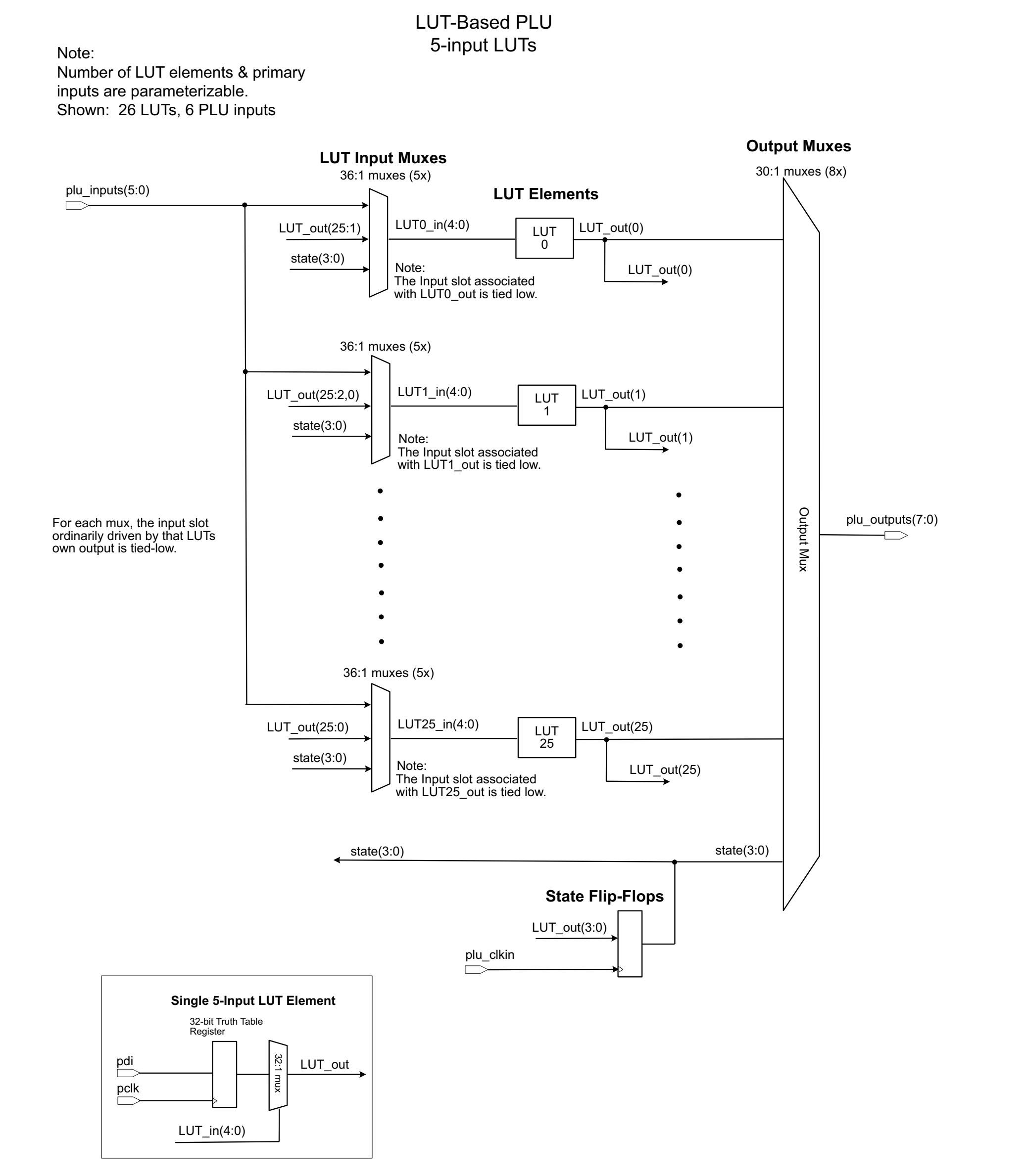

LUT-Based PLU
5-input LUTs
Note:
Number of LUT elements & primary inputs are parameterizable.
Shown: 26 LUTs, 6 PLU inputs
LUT Input Muxes
36:1 muxes (5x)
LUT Elements
Output Muxes
30:1 muxes (8x)
plu_inputs(5:0)
LUT_out(25:1)
state(3:0)
LUT0_in(4:0)
LUT 0
LUT_out(0)
Note:
The Input slot associated with LUT0_out is tied low.
LUT_out(25:2,0)
LUT1_in(4:0)
LUT 1
LUT_out(1)
The Input slot associated with LUT1_out is tied low.
For each mux, the input slot ordinarily driven by that LUTs own output is tied-low.
Output Mux
plu_outputs(7:0)
LUT_out(25:0)
LUT25_in(4:0)
LUT 25
LUT_out(25)
The Input slot associated with LUT25_out is tied low.
State Flip-Flops
LUT_out(3:0)
plu_clkin
Single 5-Input LUT Element
32-bit Truth Table Register
pdi
pclk
32:1 mux
LUT_out
LUT_in(4:0)

Fig 123. PLU block diagram

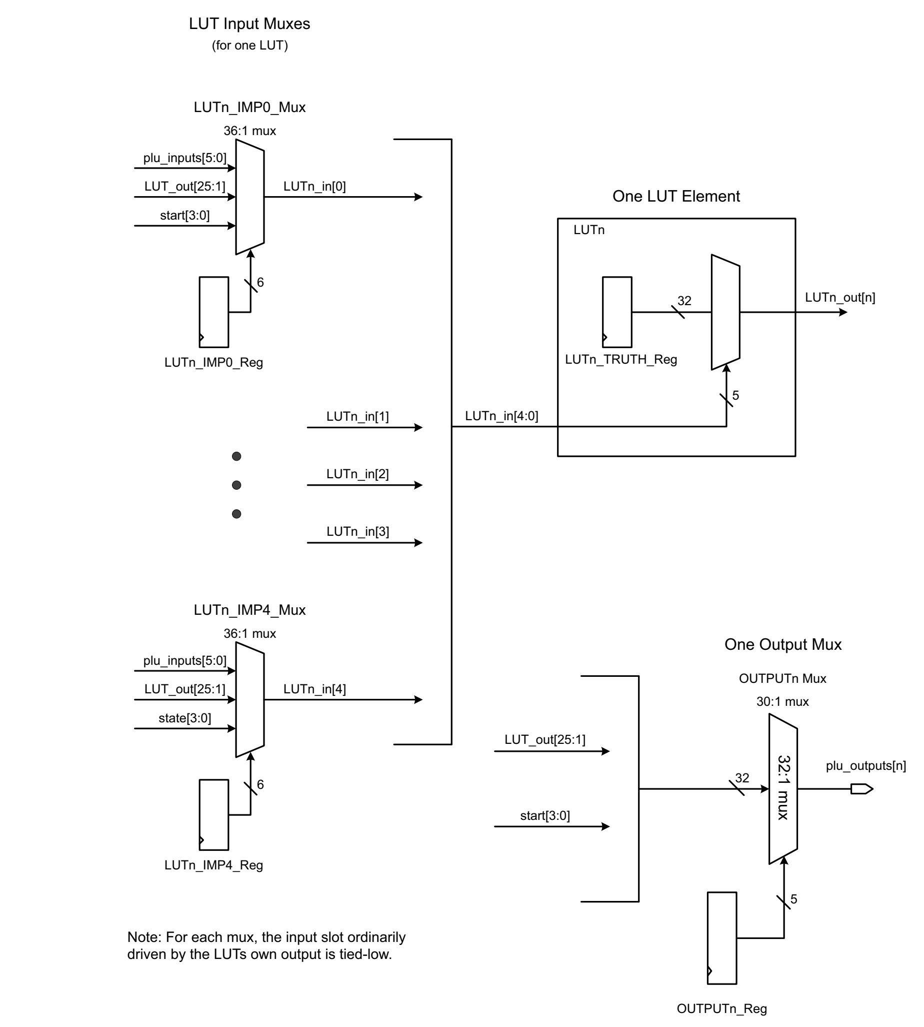


Fig 124. PLU register/multiplexing detail

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.

Note: 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 701 “Register overview: PLU (base address 0x5003D000)” 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 “LPC55S0x/LPC550x General Purpose I/O (GPIO)”.

38.4.2 Description of tool flow

The PLU programming tool suite is used to configure 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 are 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 701. 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 multiplex or 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 702. 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:** 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 703. 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 704. 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 705. 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 706. 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 706. Wake-up interrupt control for PLU (WAKEINT_CTRL, offset = 0x904) *…continued*

Bit	Symbol	Value	Description	Reset value
9:8	FILTER_MODE		Controls input of the PLU, adds 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

UM11424

Chapter 39: LPC55S0x/LPC550x 16-bit ADC controller (ADC)

Rev. 1.5 — 21 December 2023 **User manual**

39.1 How to read this chapter

The ADC controller is available on all LPC55S0x/LPC550x 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.
  - Supports 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 option 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 707 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 707. 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 708. 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_63	A-side analog channel inputs.	I
ADC0_0 - ADC0_63	B-side analog channel inputs.	I

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 709.

Table 709. VREFH selection

VREF	Mapped to
Vrefh1	vss_adc (not used).
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).

Table 710. 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	HTQFP64, HVQFN48
			1	Single Ended Conversion of CH0B	
			2	Differential Conversion CH0A-CH0B	
			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	HTQFP64, HVQFN48
			1	Single Ended Conversion of CH1B	
			2	Differential Conversion CH1A-CH1B	
			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	HTQFP64, HVQFN48
			1	Single Ended Conversion of CH2B	
			2	Differential Conversion CH2A-CH2B	
			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	HTQFP64, HVQFN48 (PIO1_0 only)
			1	Single Ended Conversion of CH3B	
			2	Differential Conversion CH3A-CH3B	
			3	Dual Single Ended Conversion CH3A & CH3B	
4	PIO1_9/ADC0_12 (referred as CH4B)"	"External STANDARD"	0	-	HTQFP64
			1	Single Ended Conversion of CH4B	
			2	-	
			3	-	

Table 710. 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
12	NA	"External STANDARD"	-	VDDA - ADC analog supply domain	NA
13	NA	"External STANDARD"	-	Internal ADC bandgap reference (1V)	NA
26	NA	"External STANDARD"	-	Temperature sensor measurement, Section 39.7.6 "Temperature sensor".	NA

39.4.3 Specific channels

Some ADC channels are used to sample specific signals.

Table 711. ADC0 pin description

Function	Connection	Description
12	VDDA	VDD3V3_ADC
13	BIAS_VREF_1V	Bias_vref_1v from aux_bias module
26	TEMP_SENSOR	Temperature sensor

39.5 General description

Figure 125 shows the ADC block diagram.

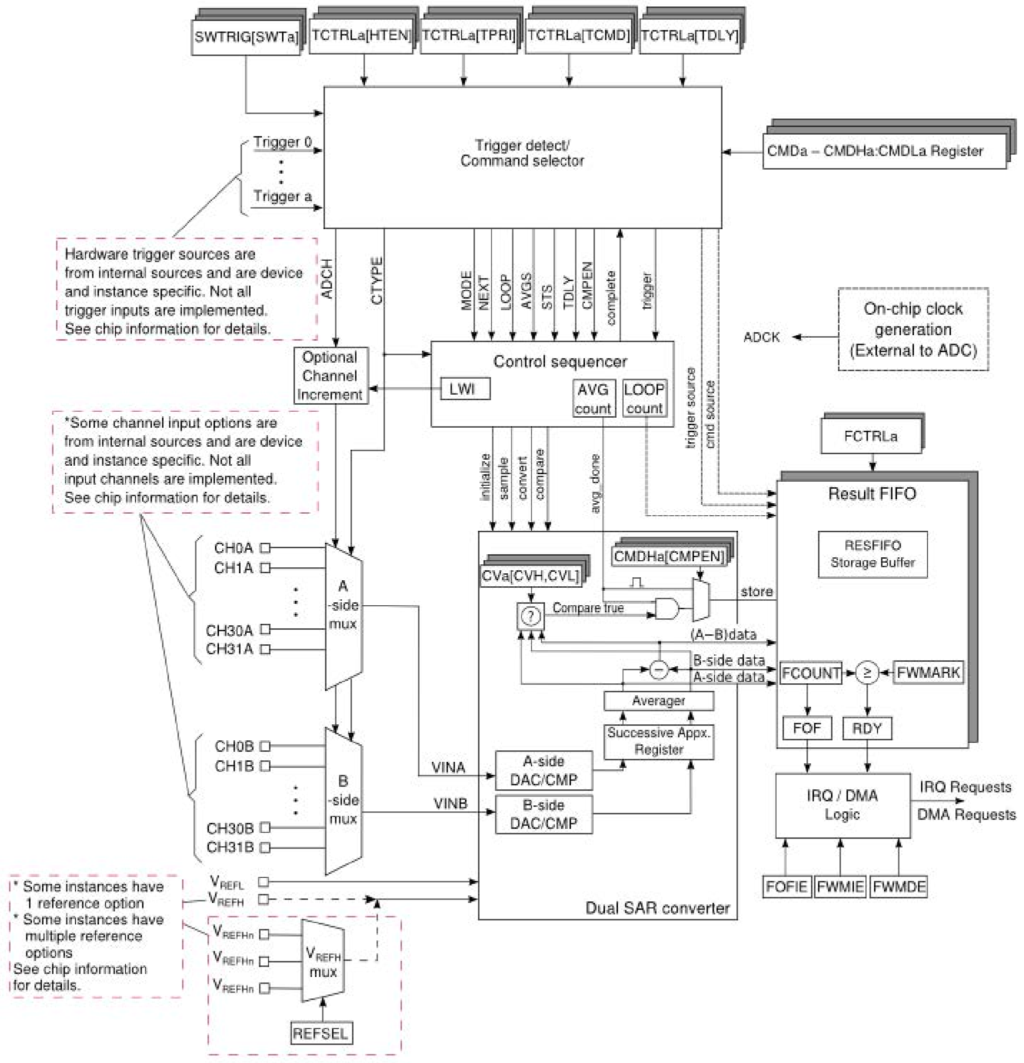

SWTRIG[SWTa]
TCTRLa[HTEN]
TCTRLa[TPRI]
TCTRLa[TCMD]
TCTRLa[TDLY]
Trigger 0
Trigger a
Trigger detect/ Command selector
CMDa – CMDHa:CMDLa Register
Hardware trigger sources are from internal sources and are device and instance specific. Not all trigger inputs are implemented. See chip information for details.
ADCH
CTYPE
MODE
NEXT
LOOP
AVGS
STS
TDLY
CMPEN
complete
trigger
ADCK
On-chip clock generation (External to ADC)
Optional Channel Increment
Control sequencer
LWI
AVG count
LOOP count
trigger source
cmd source
FCTRLa
*Some channel input options are from internal sources and are device and instance specific. Not all input channels are implemented. See chip information for details.
initialize
sample
convert
compare
avg_done
Result FIFO
RESFIFO Storage Buffer
CH0A
CH1A
CH30A
CH31A
A -side mux
CMDHa[CMPEN]
CVa[CVH,CVL]
Compare true
store
(A−B)data
B-side data
A-side data
FCOUNT
FWMARK
Averager
Successive Appx. Register
FOF
RDY
CH0B
CH1B
CH30B
CH31B
B -side mux
VINA
VINB
A-side DAC/CMP
B-side DAC/CMP
IRQ / DMA Logic
IRQ Requests
DMA Requests
FOFIE
FWMIE
FWMDE
Dual SAR converter
VREFL
VREFH
* Some instances have 1 reference option
* Some instances have multiple reference options
See chip information for details.
VREFHn
VREFH mux
REFSEL

Fig 125. 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 712. Register overview: base address 0x500A0000

Name	Access	Offset	Description	Reset value	Section
VERID	RO	0x0	Version ID register.	0x02002C0B	39.6.1
PARAM	RO	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	RO	0xF0	Gain calibration control.	0x0	39.6.14
GCC1	RO	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 712. 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 712. 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 712. 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 713. 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 713. 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 714. 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 715. 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 716. 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.	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.	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.	0x0
		0	No trigger exceptions have occurred.	
		1	A trigger exception has occurred and is pending acknowledgement.	
9	TCOMP_INT		Interrupt flag For trigger completion.	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 716. 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 717. 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 718. 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 719. 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	Higher power setting than 0b0.	
		2	Higher power setting than 0b1.	
		3	Highest power setting.	
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 719. 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 720. 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 721. 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 721. 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 722. 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 722. 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 723. 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 ($TCTRL_a$) register implements 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 show unpredictable behavior.

Table 724. 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
23:20			Reserved.	0x0

Table 724. Trigger control registers (TCTRL[0:15], offsets 0xA0 to 0xDC) *…continued*

Bit	Symbol	Value	Description	Reset value
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 725. 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 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 726. 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 727. 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 728. 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 729. 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 729. ADC command high buffer registers (CMDH[1:15], offsets 0x104 to 0x174)

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 ADCK cycles.	
		1	3.5 + 2^1 ADCK cycles; 5 ADCK cycles total sample time.	
		2	3.5 + 2^2 ADCK cycles; 7 ADCK cycles total sample time.	
		3	3.5 + 2^3 ADCK cycles; 11 ADCK cycles total sample time.	
		4	3.5 + 2^4 ADCK cycles; 19 ADCK cycles total sample time.	
		5	3.5 + 2^5 ADCK cycles; 35 ADCK cycles total sample time.	
		6	3.5 + 2^6 ADCK cycles; 67 ADCK cycles total sample time.	
		7	3.5 + 2^7 ADCK cycles; 131 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 730. 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 731 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 731. 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 732. ADC Data Result FIFO Register (RESFIFO0, offset 0x300)

Bit	Symbol	Value	Description	Reset value
15:0	D		Data result.	0x0

Table 732. 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 733. 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 733. 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 734. 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 735. 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 126 shows the sequencing of a ADC command.

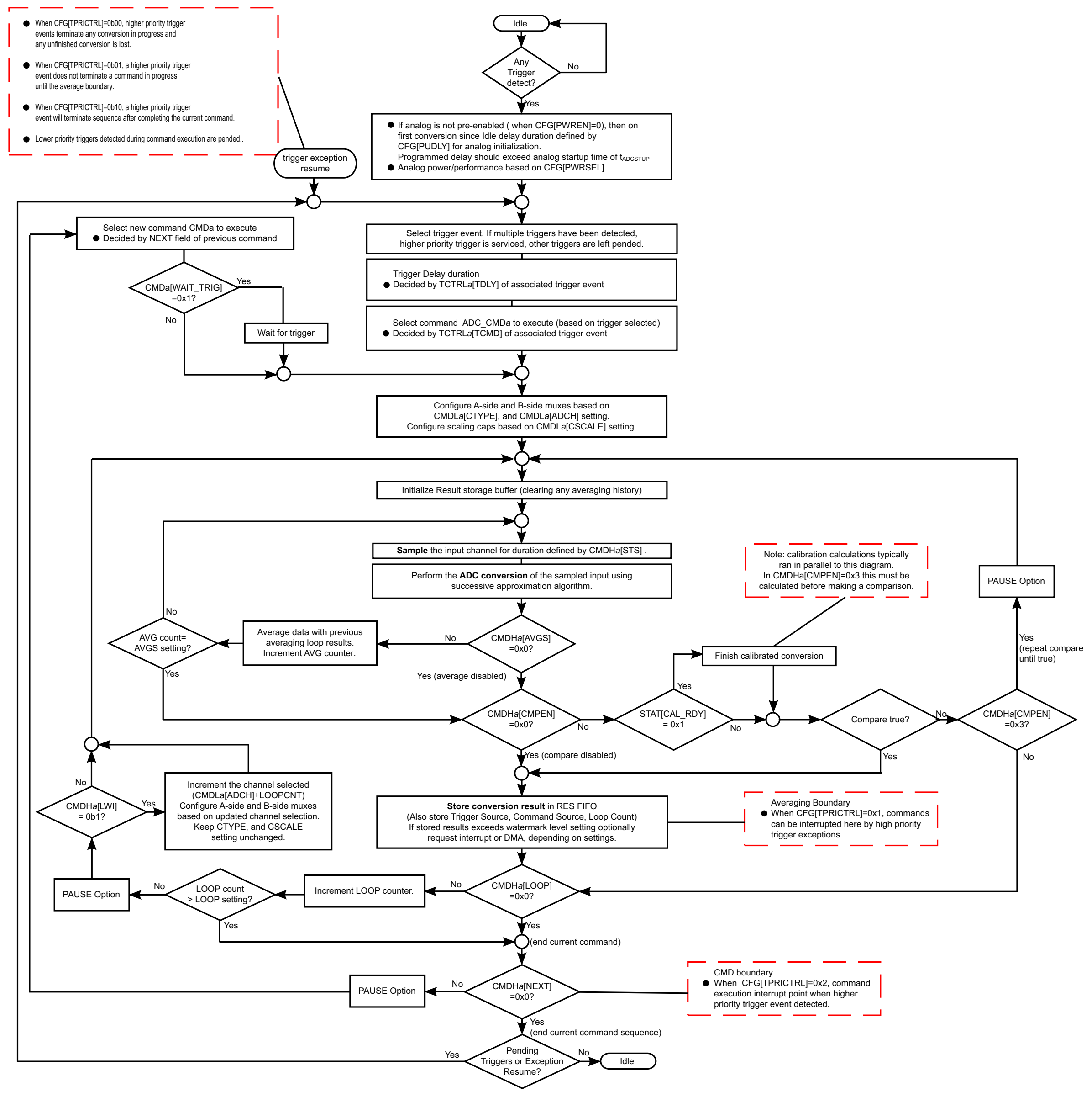

When CFG[TPRICTRL]=0b00, higher priority trigger events terminate any conversion in progress and any unfinished conversion is lost.
When CFG[TPRICTRL]=0b01, a higher priority trigger event does not terminate a command in progress until the average boundary.
When CFG[TPRICTRL]=0b10, a higher priority trigger event will terminate sequence after completing the current command.
Lower priority triggers detected during command execution are pended..
trigger exception resume
Idle
Any Trigger detect?
No
Yes
If analog is not pre-enabled (when CFG[PWREN]=0), then on first conversion since Idle delay duration defined by CFG[PUDLY] for analog initialization. Programmed delay should exceed analog startup time of t_ADCSTUP
Analog power/performance based on CFG[PWRSEL] .
Select new command CMDa to execute
Decided by NEXT field of previous command
Select trigger event. If multiple triggers have been detected, higher priority trigger is serviced, other triggers are left pended.
Trigger Delay duration
Decided by TCTRLa[TDLY] of associated trigger event
Select command ADC_CMDa to execute (based on trigger selected)
Decided by TCTRLa[TCMD] of associated trigger event
CMDa[WAIT_TRIG] =0x1?
Yes
No
Wait for trigger
Configure A-side and B-side muxes based on CMDLa[CTYPE], and CMDLa[ADCH] setting. Configure scaling caps based on CMDLa[CSCALE] setting.
Initialize Result storage buffer (clearing any averaging history)
Sample the input channel for duration defined by CMDHa[STS] .
Perform the ADC conversion of the sampled input using successive approximation algorithm.
Note: calibration calculations typically ran in parallel to this diagram. In CMDHa[CMPEN]=0x3 this must be calculated before making a comparison.
PAUSE Option
No
AVG count= AVGS setting?
Yes
Average data with previous averaging loop results. Increment AVG counter.
No
CMDHa[AVGS] =0x0?
Yes (average disabled)
Finish calibrated conversion
Yes
(repeat compare until true)
CMDHa[CMPEN] =0x0?
No
STAT[CAL_RDY] = 0x1
Yes
No
Compare true?
No
Yes
CMDHa[CMPEN] =0x3?
No
Yes (compare disabled)
No
CMDHa[LWI] = 0b1?
Yes
Increment the channel selected (CMDLa[ADCH]+LOOPCNT) Configure A-side and B-side muxes based on updated channel selection. Keep CTYPE, and CSCALE setting unchanged.
Store conversion result in RES FIFO (Also store Trigger Source, Command Source, Loop Count) If stored results exceeds watermark level setting optionally request interrupt or DMA, depending on settings.
Averaging Boundary
When CFG[TPRICTRL]=0x1, commands can be interrupted here by high priority trigger exceptions.
PAUSE Option
No
LOOP count > LOOP setting?
Yes
Increment LOOP counter.
No
CMDHa[LOOP] =0x0?
Yes
(end current command)
PAUSE Option
No
CMDHa[NEXT] =0x0?
Yes
(end current command sequence)
CMD boundary
When CFG[TPRICTRL]=0x2, command execution interrupt point when higher priority trigger event detected.
Yes
Pending Triggers or Exception Resume?
No
Idle

Fig 126. 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.

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 736.

Table 736. Power option settings

CFG[PWRSEL]	Description
0b00	Slowest speed/lowest power setting.
0b01	Faster speed/higher power setting.
0b10	Even faster speed/even higher power setting.
0b11	Fastest speed/highest power setting.

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

- Xtal clock coming directly.

The ADC target sampling rate is 1 mega sample per second. See Figure 2.

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 737. 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
7	State Counter Timer (CTIMER) ct2_mat3_out
8	State Counter Timer (CTIMER) ct3_mat3_out

Table 737. ADC hardware triggers *…continued*

Hardware trigger	Mapped to
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 126 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 127 illustrates a resync trigger sequence executing.

Note: In this example, trigger source 1 is configured to resync trigger source 0.

In Figure 127, 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.

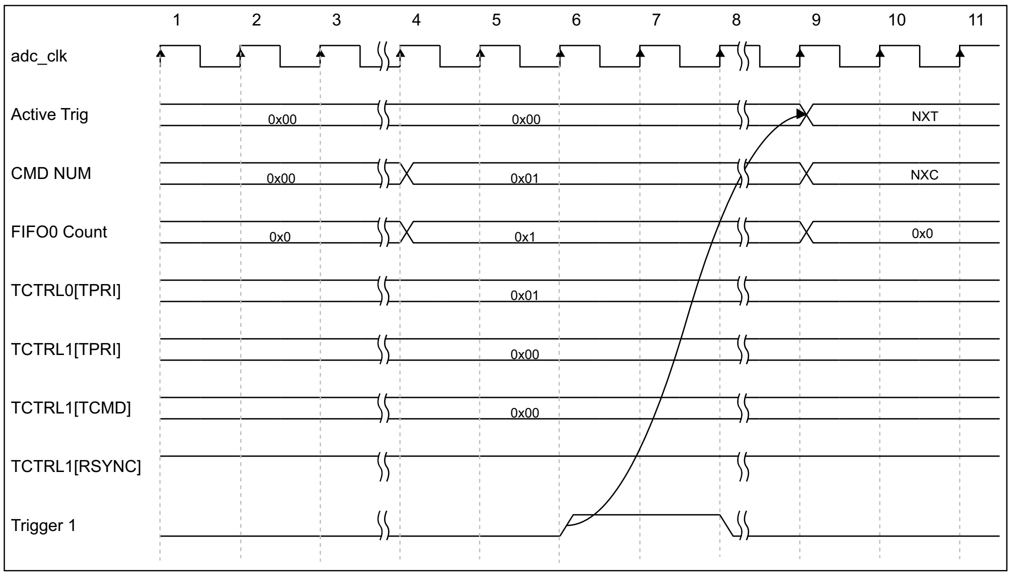


Fig 127. ADC resync example

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 128 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”>

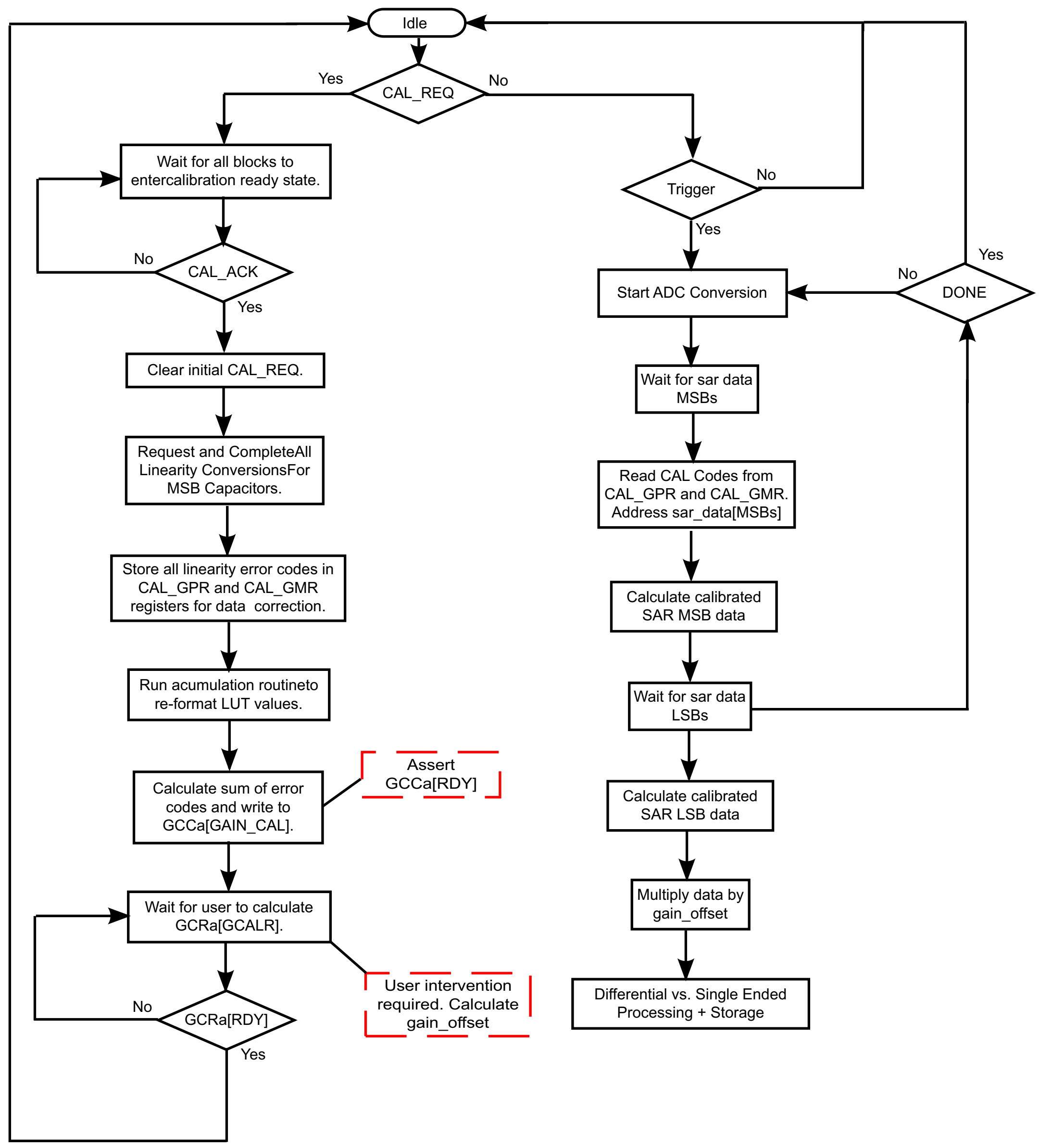

Idle
CAL_REQ
Yes
No
Wait for all blocks to entercalibration ready state.
CAL_ACK
No
Yes
Clear initial CAL_REQ.
Request and CompleteAll Linearity ConversionsFor MSB Capacitors.
Store all linearity error codes in CAL_GPR and CAL_GMR registers for data correction.
Run acumulation routineto re-format LUT values.
Calculate sum of error codes and write to GCCa[GAIN_CAL].
Assert GCCa[RDY]
Wait for user to calculate GCRa[GCALR].
User intervention required. Calculate gain_offset
GCRa[RDY]
No
Yes
Trigger
No
Yes
Start ADC Conversion
DONE
No
Yes
Wait for sar data MSBs
Read CAL Codes from CAL_GPR and CAL_GMR. Address sar_data[MSBs]
Calculate calibrated SAR MSB data
Wait for sar data LSBs
Calculate calibrated SAR LSB data
Multiply data by gain_offset
Differential vs. Single Ended Processing + Storage

Fig 128. 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 the 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] = 0x1, CMDH[LWI] = 0x0, CMDH[CMPEN] = 0x0.
4. Configure a trigger control register to associate it with the temperature sensor command: TCTRL[TCMD] = CMDT.
5. Trigger a conversion to run the command associated with TCTRL.

After running the temperature sensor command, two results are written to the FIFO selected with TCTRL[FIFO_SEL_A]. Each result corresponds to a component of the overall temperature value. To convert these two results into the temperature, software intervention is required.

To convert the two temperature sensor conversion results to the on-chip temperature value:

1. Read the two temperature sensor conversion results from the FIFO designated by TCTRL[FIFO_SEL_A].
2. The first result written by the temperature sensor command will be called Vbe1. The second result is called Vbe8.
3. An equation can be used to convert from these results into the final temperature sensor reading: A*[alpha*(Vbe8-Vbe1)/(Vbe8 + alpha*(Vbe8-Vbe1))] - B.
4. Following Alpha, A and B values are needed to achieve +/- 4 C temperature accuracy: 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.

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.

39.7.8 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 738 and Table 739.

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 $\leq$ 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 738. 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 739.

Table 739. 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 734 represents the bit widths of each register in both CAL_GAR and CAL_GBR:

Table 740. Calibration general widths

Element CAL_GxR[N]	Width (Bits)
N=0x00	11
N=0x01	12
N=0x02	13

Table 740. 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.

Chapter 40: LPC55S0x/LPC550x Analog Comparator

40.1 How to read this chapter

The analog comparator is available on all LPC55S0x/LPC550x 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 319.
- 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 “LPC55S0x/LPC550x 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 “LPC55S0x/LPC550x General Purpose I/O (GPIO)” to assign the analog comparator output to any pin on the LPC81x package.

See Table 744 to enable the analog comparator inputs and the reference voltage input.

Table 741. 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 744.

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 745 and Table 746.

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.

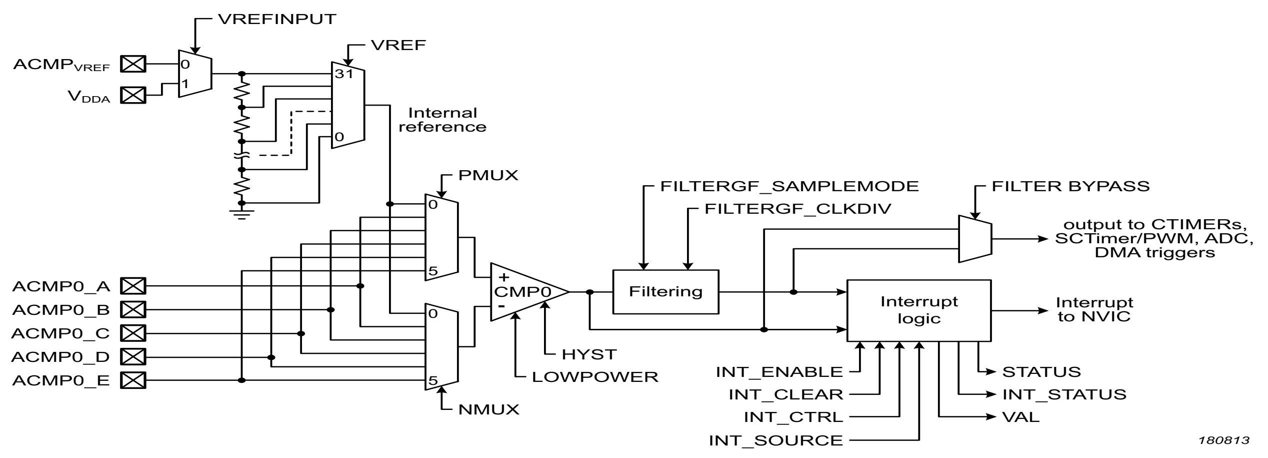


Fig 129. Comparator block diagram

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 742. 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 743. 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 744 are reserved and should be written as 0.

Table 744. 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.	
			5	PIO2_23.	
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.	
			5	Pin P2_23.	
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

Table 744. Analog comparator control register (COMP, offset = 0x50) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
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.	
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 745. 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.	

Table 745. Comparator Interrupt control (COMP_INT_CTRL, offset = 0xB10)

Bit	Symbol	Access	Value	Description	Reset value
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.	
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 746. 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

LPC55S0x/LPC550x

Chapter 41: LPC55S0x/LPC550x Controller Area Network Flexible Data

Rev. 1.5 — 21 December 2023 **User manual**

41.1 How to read this chapter

The MCAN block is available on some LPC55s0x/LPC550x devices. See Table 1 "Ordering information" for details.

41.2 Features

- Conforms to Controller Area Network (CAN) protocol version 2.0 part A, B, and ISO 11898-1
- CAN FD with up to 64 data bytes supported
- CAN error logging
- AUTOSAR support
- SAE J1939 support
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signaling on reception of High Priority Messages
- Up to 64 dedicated Receive buffers
- Up to 32 dedicated Transmit buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Message RAM is assigned to on-chip SRAM, accessible by CPU and DMA
- Shared message RAM between the two CAN FD controllers
- Programmable loop-back test mode
- Maskable module interrupts
- Power-down support
- Debug on CAN support

41.3 Basic configuration

The MCAN controller is configured using the following registers:

- Clock: In the AHBCLKCTRL1 register (Section 4.5.17), set the CAN bit to enable the function clock to the respective MCAN block being used.

  Remark: The MCAN blocks are disabled on reset (CAN = 0).

  The CAN function clock is the clock source used for the CAN. Use the CANCLKDIV register (See Section 4.5.45) to divide the main clock to reach the required CAN clock frequency if needed.

- Pins: Select the MCAN pins and pin modes through the relevant IOCON registers (See Section 15.4.2).
- Interrupts are enabled in the NVIC using the appropriate interrupt set enable register.

41.4 General description

Controller Area Network (CAN) is the definition of a high performance communication protocol for serial data communication. The MCAN controller is designed to provide a full implementation of the CAN protocol according to the CAN Specification Version 2.0 part A, B and to CAN FD Specification V.1.0.

The MCAN controller allows to build powerful local networks with low-cost multiplex wiring by supporting distributed real-time control with a very high level of security. The CAN controller consists of a CAN core, message RAM, control registers, AHB interface, and message handlers for transferring and receiving messages.

The CAN core is the CAN protocol controller and the transfer and receive shift register. The CAN core handles all ISO 11898-1 protocol functions and supports 11-bit and 29-bit identifiers.

The Tx handler transmits messages from the message RAM to the CAN Core as well as providing transmit status information. The Rx handler manages acceptance filtering and transfers received messages from the CAN core to the message RAM as well as providing receive message status information. Acceptance filtering is implemented by a combination of up to 128 filter elements where each element can be configured as a range, bit mask, or dedicated ID filter.

FigureFigure 130 shows the MCAN IP block diagram.

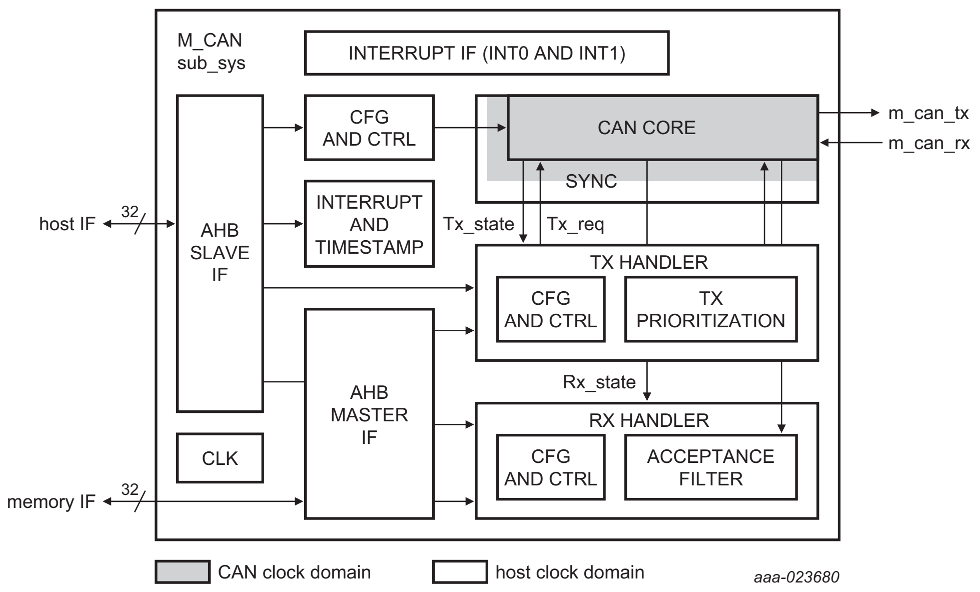


Fig 130. MCAN IP block diagram

41.5 Message RAM

For storage of Rx/Tx messages and for storage of the filter configuration, any general purpose SRAM can be used. The base address of the SRAM used for messages is determined by the value in the MRBA register that can be changed by the application.

41.5.1 Message RAM configuration

The message RAM has a width of 32 bits. The MCAN module can be configured to allocate up to 4352 words in the message RAM. It is not necessary to configure each of the sections listed in Figure 131 and there is no restriction with respect to the sequence of the sections.

When operated in CAN FD mode, the required message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx buffers, and Tx buffers via the RXESC and TXESC registers.

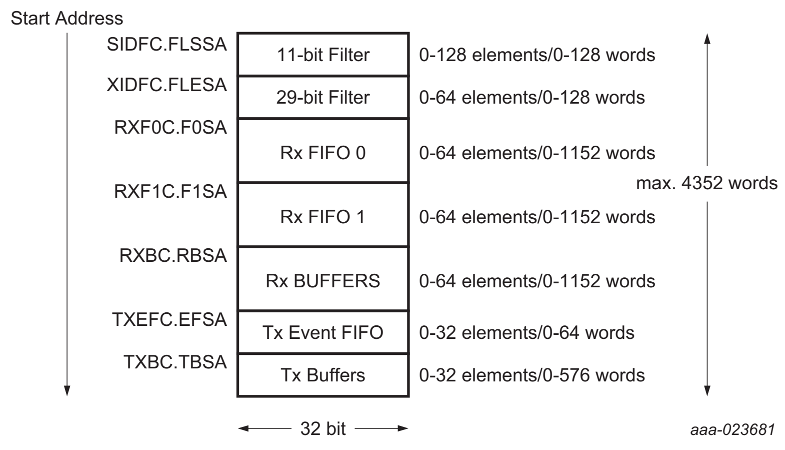


Fig 131. Message RAM configuration

When the MCAN module addresses the message RAM, it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses, that is, bits 15 to 2 are evaluated, the two least significant bits are ignored.

Remark: The MCAN does not check for erroneous configuration of the message RAM. Especially the configuration of the start addresses of different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

41.6 Pin description

Table 747. CAN pin description

Pin	Type	Description
CAN0_TD	O	MCAN transmit output
CAN0_RD	I	MCAN receive input

41.7 Register description

There is one MCAN controller on the LPC55S0x/LPC550x. Each MCAN controller contains its own set of 32-bit wide registers that are related to configuring that particular MCAN controller. See Table 748.

Table 748. Register overview: LPC55s0x/LPC550x MCAN controller (MCAN base address 0x4009 D000)

Name	Access	Offset	Description	Reset value	Section
DBTP	RW	0x00C	Data bit timing and prescaler.	0x0000 0A33	41.8.1
TEST	RW	0x010	Test register.	0	41.8.2
CCCR	RW	0x018	CC control.	0x0000 0001	41.8.3
NBTP	RW	0x01C	Nominal bit timing and prescaler.	0x0600 0A03	41.8.4
TSCC	RW	0x020	Timestamp counter configuration.	0	41.8.5
TSCV	RO	0x024	Timestamp counter value.	0	41.8.6
TOCC	RW	0x028	Timeout counter configuration.	0xFFFF 0000	41.8.7
TOCV	RO	0x02C	Timeout counter value.	0x0000 FFFF	41.8.8
ECR	RO	0x040	Error counter.	0	41.8.9
PSR	RO	0x044	Protocol status.	0x0000 0707	41.8.10
TDCR	RW	0x048	Transmitter delay compensator.	0	41.8.11
IR	RW	0x050	Interrupt.	0	41.8.12
IE	RW	0x054	Interrupt enable.	0	41.8.13
ILS	RW	0x058	Interrupt line select.	0	41.8.14
ILE	RW	0x05C	Interrupt line enable.	0	41.8.15
GFC	RW	0x080	Global filter configuration.	0	41.8.16
SIDFC	RW	0x084	Standard ID filter configuration.	0	41.8.17
XIDFC	RW	0x088	Extended ID filter configuration.	0	41.8.18
XIDAM	RW	0x090	Extended ID and mask.	0x1FFF FFFF	41.8.19
HPMS	RO	0x094	High priority message status.	0	41.8.20
NDAT1	RW	0x098	New data 1.	0	41.8.21
NDAT2	RW	0x09C	New data 2.	0	41.8.22
RXF0C	RW	0x0A0	Rx FIFO 0 configuration.	0	41.8.23
RXF0S	RO	0x0A4	Rx FIFO 0 status.	0	41.8.24
RXF0A	RW	0x0A8	Rx FIFO 0 acknowledge.	0	41.8.25
RXBC	RW	0x0AC	Rx buffer configuration.	0	41.8.26
RXF1C	RW	0x0B0	Rx FIFO 1 configuration.	0	41.8.27
RXF1S	RO	0x0B4	Rx FIFO 1 status.	0	41.8.28
RXF1A	RW	0x0B8	Rx FIFO 1 acknowledge.	0	41.8.29
RXESC	RW	0x0BC	Rx buffer and FIFO element size configuration.	0	41.8.30
TXBC	RW	0x0C0	Tx buffer configuration.	0	41.8.31
TXFQS	RW	0x0C4	Tx FIFO/queue status.	0	41.8.32
TXESC	RW	0x0C8	Tx buffer element size configuration.	0	41.8.33
TXBRP	RO	0x0CC	Tx buffer request pending.	0	41.8.34
TXBAR	RW	0x0D0	Tx buffer add request.	0	41.8.35
TXBCR	RW	0x0D4	Tx buffer cancellation request.	0	41.8.36

Table 748. Register overview: LPC55s0x/LPC550x MCAN controller (MCAN base address 0x4009 D000) *…continued*

Name	Access	Offset	Description	Reset value	Section
TXBTO	RO	0x0D8	Tx buffer transmission occurred.	0	41.8.37
TXBCF	RO	0x0DC	Tx buffer cancellation finished.	0	41.8.38
TXBTIE	RW	0x0E0	Tx buffer transmission interrupt enable.	0	41.8.39
TXBCIE	RW	0x0E4	Tx buffer cancellation finished interrupt enable.	0	41.8.40
TXEFC	RW	0x0F0	Tx event FIFO configuration.	0	41.8.41
TXEFS	RO	0x0F4	Tx event FIFO status.	0	41.8.42
TXEFA	RW	0x0F8	Tx event FIFO acknowledge.	0	41.8.43
MRBA	RW	0x200	CAN message RAM base address.	0	41.8.44
ETSCC	RW	0x400	External timestamp counter configuration.	0	41.8.45
ETSCV	RW	0x600	External timestamp counter value.	0	41.8.46

41.8 CAN protocol register description

41.8.1 Data bit timing and prescaler register

Write access to the DBTP register is enabled by setting bits CCE and INIT in the CCCR register.

The nominal bit time for CAN is determined by the number of time quanta per bit, and the time each time quantum represents.The time quantum (t_q) may be programmed in the range of 1 to 32 MCAN clock periods:

T_q = (DBRP + 1) MCAN clock periods

A single CAN bit time is made up from 4 segments:

Segment SYNC_SEG is not programmable and is fixed at one time quantum.

Segments PROP_SEG and PHASE_SEG1 are combined in a single bit field, and are programmable in the range of 1 to 16 time quanta. The sample point is set right after PHASE_SEG1.

Segment PHASE_SEG2 is programmable in the range of 1 to 16 time quanta.

The length of the bit time (in time quanta) is the combined value of these 4 segments:

[SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] t_q = [1 + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] t_q

Therefore, the CAN bit time may be programmed in the range of 3 to 33 time quanta.

The combined value of segments PROP_SEG and PHASE_SEG1 can be programmed using bit field DTSEG1, and segment PHASE_SEG2 can be programmed using bit field DTSEG2.

The CAN hardware interprets these bit fields as the programmed value+1, so the length of the bit time (in time quanta) can be calculated from the programmed values as:

[1 + DTSEG1 + 1 + DTSEG2 + 1] t_q = [DTSEG1 + DTSEG2 + 3] t_q

The CAN bit time may be programmed in the range of 4 to 49 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 MCAN clock periods.

T_q = (DBRP + 1) mtq.

DTSEG1 is the sum of prop_seg and phase_Seg1. DTSEG2 is phase_seg2. Therefore, the length of the bit time is:

[DTSEG1 + DTSEG2 + 3] t_q for programmed values, or,

[sync_seg + prop_seg + phase_seg1 + phase_seg2] t_q for functional values.

The information processing time (IPT) is zero, meaning that the data for the next bit is available the first clock edge after the sample point.

Table 749. Data bit timing and prescaler register (DBTP, offset 0x00C) bit description

Table 750.

Bit	Symbol	Value	Description	Reset value
3:0	DSJW	-	Data (re)synchronization jump width. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 15. Limits by how many time quanta a bit period may be extended or shortened because of re-synchronization.	0x3
7:4	DTSEG2	-	Data time segment after sample point. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 15.	0x3
12:8	DTSEG1	-	Data time segment before sample point. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 31.	0xA
15:13	-	-	Reserved.	-
20:16	DBRP	-	Data bit rate prescaler. The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.Valid values are 0 to 31.	0
22:21	-	-	Reserved.	-
23	TDC		Transmitter delay compensation.	0
		0	Transmitter delay compensation disabled	
		1	Transmitter delay compensation enabled	
31:24	-	-	Reserved	-

Remark: With a MCAN clock of 8 MHz, the reset value of DBTP register will configure the MCAN for a data phase bit rate of 500 kb/s.

Remark: The bit rate configured for the CAN FD data phase via the DBTP register must be higher or equal to the bit rate configured for the arbitration phase via the NBTP register.

41.8.2 Test register

Write access to the CAN test register is enabled by setting the TEST bit in the CCCR register. All test register functions are set to their reset values when the TEST bit is reset.

The different test functions can be combined, but when TX bits are programmed with a value that is not 0x0, the message transfer is disturbed.

Table 751. Test register (TEST, offset 0x010) bit description

Bit	Symbol	Value	Description	Reset value
3:0	-	-	Reserved.	-
4	LBCK		Loop back mode.	0
		0	Loop back mode is disabled	
		1	Loop back mode is enabled	

Table 751. Test register (TEST, offset 0x010) bit description

Bit	Symbol	Value	Description	Reset value
6:5	TX		Control of transmit pin.	0
		0x0	Controller. Level of CAN_TXD is controlled by CAN controller. This is the value at reset.	
		0x1	Sample point. The sample point can be monitored at the CAN_TXD.	
		0x2	Low. CAN_TXD pin is driven LOW/dominant.	
		0x3	High. CAN_TXD is driven HIGH/recessive.	
7	RX		Monitors the actual value of the CAN_RXD.	0
		0	Dominant. The CAN bus is dominant (CAN_RXD = 0).	
		1	Recessive. The CAN bus is recessive (CAN_RXD = 1).	
31:8	-	-	Reserved.	-

41.8.3 Control register

The MCAN module has a mechanism to synchronize the two clock domains within itself, which may cause a delay before the value written to the INIT bit can be read back. Due to this, it is recommended to read back the value of the INIT bit and confirm that it has been accepted before writing a new value to the INIT bit.

Table 752. Control register (CCCR, offset 0x018) bit description

Bit	Symbol	Value	Description	Reset value
0	INIT		Initialization.	1
		0	Normal operation.	
		1	Initialization is started.	
1	CCE		Configuration change enable.	0
		0	No write access. The CPU has no write access to the protected configuration registers.	
		1	Write access. The CPU has write access to the protected configuration registers.	
2	ASM		Restricted operational mode.	0
		0	Normal CAN operation.	
		1	Restricted operation mode active.	
3	CSA		Clock stop acknowledge.	0
		0	No clock stop acknowledged.	
		1	MCAN may be set in power down by stopping the internal MCAN clocks.	
4	CSR		Clock stop request.	0
		0	No clock stop is requested.	
		1	Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reaches idle.	
5	MON		Bus monitoring mode.	0
		0	Bus monitoring mode is disabled.	
		1	Bus monitoring mode is enabled.	
6	DAR		Disable automatic retransmission.	0
		0	Automatic retransmission of messages not transmitted successfully enabled.	
		1	Automatic retransmission disabled.	

Table 752. Control register (CCCR, offset 0x018) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
7	TEST		Test mode enable.	0
		0	Normal operation.	
		1	Test mode is enabled.	
8	FDOE		CAN FD operation enable.	0
		0	CAN FD operation is disabled.	
		1	CAN FD operation is enabled.	
9	BRSE		When CAN FD operation is disabled, this bit is not evaluated.	0
		0	Bit rate switching for transmissions is disabled.	
		1	Bit rate switching for transmission is enabled.	
11:10	-	-	Reserved.	-
12	PXHD		Protocol exception handling disable. When protocol exception handling is disabled, the MCAN module will transmit an error frame hen it detects a protocol exception condition.	0
		0	Protocol exception handling is enabled.	
		1	Protocol exception handling is disabled.	
13	EFBI		Edge filtering during bus integration.	0
		0	Edge filtering is disabled.	
		1	Two consecutive dominant quanta required to detect an edge for hard synchronization.	
14	TXP		Transmit pause.	0
		0	Transmit pause is disabled.	
		1	Transmit pause is enabled.	
15	NISO		Non ISO operation. If this bit is set, the MCAN module uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.	0
		0	CAN FD frame format will follow according to ISO11898-1.	
		1	CAN FD frame format will follow according to Bosch CAN FD Specification V1.0.	
31:16	-	-	Reserved.	-

41.8.4 Nominal bit timing and prescaler register

Write access to the NBTP register is enabled by setting the CCE and INIT bits in the CCCR register.

The CAN bit time may be programmed in the range of 4 to 385 time quanta. The CAN time quantum may be programmed in the range of 1 to 152 MCAN clock periods.

T_q = (DBRP + 1) MCAN clock periods

NTSEG1 is the sum of prop_seg and phase_Seg1. DTSEG2 is phase_seg2. Therefore, the length of the bit time is:

[NTSEG1 + NTSEG2 + 3] t_q for programmed values, or,

[sync_seg + prop_seg + phase_seg1 + phase_seg2] t_q for functional values.

The information processing time (IPT) is zero, meaning that the data for the next bit is available upon the first clock edge after the sample point.

Table 753. Nominal bit timing and prescaler register (NBTP, offset 0x01C) bit description

Bit	Symbol	Description	Reset value
6:0	NTSEG2	Nominal time segment after sample point. The actual interpretation by the hardware for this value is such that one more than the value programmed here is used. Valid values are 0 to 127.	0x3
7	-	Reserved.	-
15:8	NTSEG1	Nominal time segment before sample point. The actual interpretation by that hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 255.	0xA
24:16	NBRP	Nominal bit rate prescaler. The value by which the oscillator frequency is divided for generating the bit time quanta. The actual interpretation by that hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 511.	0
31:25	NSJW	Nominal (re)synchronization jump width. The actual interpretation by that hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 127.	0x3

Remark: With a MCAN clock of 8 MHz, the reset value of NBTP register will configure the MCAN for a data phase bit rate of 500 kb/s.

41.8.5 Timestamp counter configuration register

Table 754. Timestamp counter configuration register (TSCC, offset 0x020) bit description

Bit	Symbol	Value	Description	Reset value
1:0	TSS		Timestamp select.	0
		0x0	Timestamp counter value static at 0x0000.	
		0x1	Timestamp counter value incremented according to TCP bits.	
		0x2	External timestamp counter value used.	
		0x3	Timestamp counter value static at 0x0000.	
15:2	-	-	Reserved.	
19:16	TCP	-	Timestamp counter prescaler. Configures the timestamp and timeout counters time unit in multiple of CAN bit times. The actual interpretation by that hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 15.	0
31:20	-	-	Reserved.	-

41.8.6 Timestamp counter value register

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). The timestamp counter will increment depending on the TSS bits in the TSCC register. An overflow will set the TSW interrupt flag in the IR register.

Table 755. Timestamp counter value register (TSCV, offset 0x024) bit description

Bit	Symbol	Description	Reset value
15:0	TSC	Timestamp counter.	0
31:16	-	Reserved.	-

41.8.7 Timeout counter configuration register

Table 756. Timeout counter configuration register (TOCC, offset 0x028) bit description

Bit	Symbol	Value	Description	Reset value
0	ETOC		Enable timeout counter.	0
		0	Timeout counter is disabled.	
		1	Timeout counter is enabled.	
2:1	TOS		Timeout select. When the timeout counter is operating in continuous mode, a write to the TOCV register presets the counter to the value configured in the TOP bits of the TOCC register and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by the TOP bits. Down-counting is started when the first FIFO element is stored.	0
		0x0	Continuous operation.	
		0x1	Timeout is controlled by Tx event FIFO.	
		0x2	Timeout is controlled by Rx FIFO 0.	
		0x3	Timeout is controlled by Rx FIFO 1.	
15:3	-	-	Reserved.	-
31:16	TOP	-	Timeout Period. This register holds the start value of the timeout counter to configure the timeout period. This counter counts down.	0xFFFF

41.8.8 Timeout counter value register

Table 757. Timeout counter value register (TOCV, offset 0x02C) bit description

Bit	Symbol	Description	Reset value
15:0	TOC	Timeout counter. The timeout counter is decremented in multiples of CAN bit times depending on the configuration of the TCP bits in the TSCC register. When decremented to zero, the TOO interrupt flag is set in the IR register and the timeout counter is stopped. Start and reset conditions are configured by the TOS bits in the TOC register.	0xFFFF
31:16	-	Reserved.	-

41.8.9 Error counter register

Table 758. Error counter register (ECR, offset 0x040) bit description

Bit	Symbol	Value	Description.	Reset value
7:0	TEC	-	Transmit error counter. Current value of the transmit error counter. Current value of the transmit error counter. Valid values are between 0 and 255.	0
14:8	REC	-	Receive error counter. Current value of the receive error counter. Valid values are between 0 and 127.	0
15	RP		Receive error passive.	0
		0	Below error level. The receive counter is below the error passive level of 128.	
		1	At error level. The receive counter has reached the error passive level of 128.	
23:16	CEL	-	CAN error logging. CEL is incremented when TEC or REC is incremented. The counter stops at 0xFF and the next occurrence of a CAN protocol error will set the ELO interrupt flag in the IR register. This counter is reset whenever a read access to these bits is made.	0
31:24	-	-	Reserved.	-

Remark: When the ASM bit in the CCCR register is set, the CAN protocol controller does not increment the TEC and REC bits when a CAN protocol error is detected, but the CEL bits are still incremented.

41.8.10 Protocol status register

Table 759. Protocol status register (PSR, offset 0x044) bit description

Bit	Symbol	Value	Description	Reset value
2:0	LEC		Last error code. These bits indicate the type of the last error to occur on the CAN bus. This bit field will be cleared when a message has been transferred without error. The bits in this bit field will be set upon a read access.	0x7
		0x0	No error: No error has occurred since LEC bits has been reset by successful reception or transmission.	
		0x1	Stuff error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.	
		0x2	Form error: A fixed format part of a received frame has the wrong format.	
		0x3	AckError: The message transmitted by the M_CAN was not acknowledged by another node.	
		0x4	Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.	
		0x5	Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).	
		0x6	CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.	
		0x7	NoChange: Any read access to the protocol status register re-initializes the LEC bits to 0x7. When the LEC bits equal the value 0x7, no CAN bus event was detected since the last CPU read access to the protocol status register.	
4:3	ACT		Activity. This register monitors the MCAN communication state.	0
		0x0	Synchronizing – node is synchronizing on CAN communication.	
		0x1	Idle – node is neither receiver nor transmitter.	
		0x2	Receiver – node is operating as receiver.	
		0x3	Transmitter – node is operating as transmitter.	
5	EP		Error passive.	0
		0	The MCAN is in Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected.	
		1	The MCAN is in the Error_Passive state.	
6	EW		Warning status.	0
		0	Both error counters are below the Error_Warning limit of 96.	
		1	At least one of error counter has reached the Error_Warning limit of 96.	
7	BO		Bus off status.	0

Table 759. Protocol status register (PSR, offset 0x044) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
10:8	DLEC	-	Data phase last error code. Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC bits. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error. The bits in this bit field will be set upon a read access.	0x7
11	RESI		ESI flag of the last received CAN FD message. This bit is set together with RFDF bits, independent of acceptance filtering. This bit field will be set upon a read access.	0
		0	Last received CAN FD message did not have its ESI flag set.	
		1	Last received CAN FD message had its ESI flag set.	
12	RBRS		BRS flag of last received CAN FD message. This bit is set together with RFDF bits, independent of acceptance filtering. This bit field will be set upon a read access.	0
		0	Last received CAN FD message did not have its BRS flag set.	
		1	Last received CAN FD message had its BRS flag set.	
13	RFDF		Received a CAN FD message. This bit is set independent of acceptance filtering. This bit field will be set on a read access.	0
		0	No CAN FD message received since the last CPU reset.	
		1	Message in CAN FD format with FDF flag set has been received.	
14	PXE		Protocol exception event. This bit field will be set upon a read access.	0
		0	No protocol exception event occurred since last read access.	
		1	Protocol exception event occurred.	
15	-	-	Reserved.	-
22:16	TDCV	-	Transmitter delay compensation value. Position of the secondary sample point, defined by the sum of the measured delay from m_can_tx and m_can_rx and TDCO bits in the TDCR register. The SSP position in the data phase is the number of MCAN clock periods between the start of a transmitted bit and secondary sample point. Valid values are 0 to 127 MCAN clock periods.	
31:23	-	-	Reserved.	-

Remark: When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in DLEC bits instead of LEC bits. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.

Remark: The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting the INIT bit in the CCCR register. If the device goes Bus_Off, it will set INIT bit of its own accord, stopping all bus activities. Once the INIT bit has been cleared, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of INIT bit, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to the LEC bits to check whether the CAN bus is stuck at dominant or continuously disturbed, and to monitor the Bus_Off recovery sequence. The REC bits in the ECR register is used to count these sequences.

41.8.11 Transmitter delay compensation register

Table 760. Transmitter delay compensation register (TDCR, offset 0x044) bit description

Bit	Symbol	Value	Description	Reset value
6:0	TDCF	-	Transmitter delay compensation filter window length. Defines the minimum value for the SSP position, dominant edges on m_can_rx that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF bits are configured to a value greater than TDCO bits. Valid values are 0 to 127 MCAN clock periods.	0
7	-	-	Reserved.	-
14:8	TDCO	-	Transmitter delay compensation offset. Offset value defining the distance between the measured delay from m_can_tx to m_can_rx and the secondary sample point. Valid values are 0 to 127 MCAN clock periods.	0
31:15	-	-	Reserved.	-

41.8.12 Interrupt register

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register. The configuration of the IE register controls whether an interrupt is generated. The configuration of the ILS register controls on which interrupt line an interrupt is signaled.

Table 761. Interrupt register (IR, offset 0x050) bit description

Bit	Symbol	Value	Description	Reset value
0	RF0N		Rx FIFO 0 new message.	0
		0	No new message written to Rx FIFO 0.	
		1	New message written to Rx FIFO 0.	
1	RF0W		Rx FIFO 0 watermark reached.	0
		0	Rx FIFO 0 fill level below watermark.	
		1	Rx FIFO 0 fill level reached watermark.	
2	RF0F		Rx FIFO 0 full.	0
		0	Rx FIFO 0 not full.	
		1	Rx FIFO 0 full.	
3	RF0L		Rx FIFO 0 message lost.	0
		0	No Rx FIFO 0 message lost.	
		1	Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero.	
4	RF1N		Rx FIFO 1 new message.	0
		0	No new message written to Rx FIFO 1.	
		1	New message written to Rx FIFO 1.	
5	RF1W		Rx FIFO 1 watermark reached.	0
		0	Rx FIFO 1 fill level below watermark.	
		1	Rx FIFO 1 fill level reached watermark.	
6	RF1F		Rx FIFO 1 full.	0
		0	Rx FIFO 1 not full.	
		1	Rx FIFO 1 full.	

Table 761. Interrupt register (IR, offset 0x050) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
7	RF1L		Rx FIFO 1 message lost.	0
		0	No Rx FIFO 1 message lost	
		1	Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero.	
8	HPM		High priority message.	0
		0	No high priority message received.	
		1	High priority message received.	
9	TC		Transmission completed.	0
		0	No transmission completed.	
		1	Transmission completed.	
10	TCF		Transmission cancellation finished.	0
		0	No transmission cancellation finished.	
		1	Transmission cancellation finished.	
11	TFE		Tx FIFO empty.	0
		0	Tx FIFO non-empty.	
		1	Tx FIFO empty.	
12	TEFN		Tx event FIFO new entry.	0
		0	Tx event FIFO unchanged.	
		1	Tx Handler wrote Tx event FIFO element.	
13	TEFW		Tx event FIFO watermark reached.	0
		0	Tx event FIFO fill level below watermark.	
		1	Tx event FIFO fill level reached watermark.	
14	TEFF		Tx event FIFO full.	0
		0	Tx event FIFO not full.	
		1	Tx event FIFO full.	
15	TEFL		Tx event FIFO element lost.	0
		0	No Tx event FIFO element lost.	
		1	Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size zero.	
16	TSW		Timestamp wraparound.	0
		0	No timestamp counter wraparound.	
		1	Timestamp counter wrapped around.	

Table 761. Interrupt register (IR, offset 0x050) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
17	MRAF		Message RAM access failure. The flag is set when the Rx Handler meets either of the following criteria: The Rx handler has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message. The Rx handler was not able to write a message to the Message RAM and the message storage is aborted. In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx buffer is not set, a partly stored message is overwritten when the next message is stored to this location. The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the MCAN is switched into Restricted Operation Mode. To leave Restricted Operation Mode, the ASM bit in the CCCR must be cleared.	0
		0	No message RAM access failure occurred.	
		1	Message RAM access failure occurred.	
18	TOO		Timeout occurred.	0
		0	No timeout.	
		1	Timeout reached.	
19	DRX		Message stored in dedicated Rx buffer.	0
		0	No Rx buffer updated.	
		1	At least one received message stored into an Rx buffer.	
20	BEC		Bit error corrected. Message RAM bit error detected and corrected. Controlled by input signal m_can_aeim_berr[0] generated by an optional external parity / ECC logic attached to the message RAM.	0
		0	No bit error detected when reading from message RAM.	
		1	Bit error detected and corrected (example, ECC).	
21	BEU		Bit error uncorrected. Message RAM bit error detected, uncorrected. Controlled by input signal m_can_aeim_berr[1] generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets INIT bit in the CCCR register to 1. This is done to avoid transmission of corrupted data.	0
		0	No bit error detected when reading from message RAM.	
		1	Bit error detected, uncorrected (example, parity logic).	
22	ELO		Error logging overflow.	0
		0	CAN error logging counter did not overflow.	
		1	Overflow of CAN error logging counter occurred.	
23	EP		Error passive.	0
		0	Error_Passive status unchanged.	
		1	Error_Passive status changed.	
24	EW		Warning status.	0
		0	Error_Warning status unchanged.	
		1	Error_Warning status changed.	

Table 761. Interrupt register (IR, offset 0x050) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
25	BO		Bus_Off Status.	0
		0	Bus_Off status unchanged.	
		1	Bus_Off status changed.	
26	WDI		Watchdog interrupt.	0
		0	No message RAM watchdog event occurred.	
		1	Message RAM watchdog event due to missing READY.	
27	PEA		Protocol error in arbitration phase.	0
		0	No protocol error in arbitration phase.	
		1	Protocol error in arbitration phase detected.	
28	PED		Protocol error in data phase.	0
		0	No protocol error in data phase.	
		1	Protocol error in data phase detected.	
29	ARA		Access to reserved address.	0
		0	No access to reserved address occurred.	
		1	Access to reserved address occurred.	
31:30	-	-	Reserved.	-

41.8.13 Interrupt enable register

The setting in the interrupt enable register determines which status changes in the interrupt register will be signaled on an interrupt line.

Table 762. Interrupt enable register (IE, offset 0x054) bit description

Bit	Symbol	Value	Description	Reset value
0	RF0NE		Rx FIFO 0 new message interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
1	RF0WE		Rx FIFO 0 watermark reached interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
2	RF0FE		Rx FIFO 0 full interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
3	RF0LE		Rx FIFO 0 message lost interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
4	RF1NE		Rx FIFO 1 new message interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	

Table 762. Interrupt enable register (IE, offset 0x054) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
5	RF1WE		Rx FIFO 1 watermark reached interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
6	RF1FE		Rx FIFO 1 full interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
7	RF1LE		Rx FIFO 1 message lost interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
8	HPME		High priority message interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
9	TCE		Transmission completed interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
10	TCFE		Transmission cancellation finished interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
11	TFEE		Tx FIFO empty interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
12	TEFNE		Tx event FIFO new entry interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
13	TEFWE		Tx event FIFO watermark reached interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
14	TEFFE		Tx event FIFO full interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
15	TEFLE		Tx event FIFO element lost interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
16	TSWE		Timestamp wraparound interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
17	MRAFE		Message RAM access failure interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	

Table 762. Interrupt enable register (IE, offset 0x054) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
18	TOOE		Timeout occurred interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
19	DRXE		Message stored in dedicated Rx buffer interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
20	BECE		Bit error corrected interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
21	BEUE		Bit error uncorrected interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
22	ELOE		Error logging overflow interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
23	EPE		Error passive interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
24	EWE		Warning status interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
25	BOE		Bus_Off Status interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
26	WDIE		Watchdog interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
27	PEAE		Protocol error in arbitration phase interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
28	PEDE		Protocol error in data phase interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
29	ARAE		Access to reserved address interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
31:30	-	-	Reserved.	-

41.8.14 Interrupt line select register

The interrupt line select register assigns an interrupt generated by a specific interrupt flag from the interrupt register. For interrupt generation, the respective interrupt line has to be enabled via the interrupt line enable register.

Table 763. Interrupt line select register (ILS, offset 0x058) bit description

Bit	Symbol	Value	Description	Reset value
0	RF0NL		Rx FIFO 0 new message interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
1	RF0WL		Rx FIFO 0 watermark reached interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
2	RF0FL		Rx FIFO 0 full interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
3	RF0LL		Rx FIFO 0 message lost interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
4	RF1NL		Rx FIFO 1 new message interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
5	RF1WL		Rx FIFO 1 watermark reached interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
6	RF1FL		Rx FIFO 1 full interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
7	RF1LL		Rx FIFO 1 message lost interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
8	HPML		High priority message interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
9	TCL		Transmission completed interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
10	TCFL		Transmission cancellation finished interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
11	TFEL		Tx FIFO empty interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	

Table 763. Interrupt line select register (ILS, offset 0x058) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
12	TEFNL		Tx event FIFO new entry interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
13	TEFWL		Tx event FIFO watermark reached interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
14	TEFFL		Tx event FIFO full interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
15	TEFLL		Tx event FIFO element lost interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
16	TSWL		Timestamp wraparound interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
17	MRAFL		Message RAM access failure interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
18	TOOL		Timeout occurred interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
19	DRXL		Message stored in dedicated Rx buffer interrupt line	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
20	BECL		Bit error corrected interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
21	BEUL		Bit error uncorrected interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
22	ELOL		Error logging overflow interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
23	EPL		Error passive interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
24	EWL		Warning status interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	

Table 763. Interrupt line select register (ILS, offset 0x058) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
25	BOL		Bus_Off Status interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
26	WDIL		Watchdog interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
27	PEAL		Protocol error in arbitration phase interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
28	PEDL		Protocol error in data phase interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
29	ARAL		Access to reserved address interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
31:30	-	-	Reserved.	-

41.8.15 Interrupt line enable register

The interrupt line enable register is used to enable or disable the two interrupt lines available.

Table 764. Interrupt line enable register (ILE, offset 0x05C) bit description

Bit	Symbol	Value	Description	Reset value
0	EINT0		Enable interrupt line 0.	0
		0	Interrupt line to MCANx_INT0 is disabled	
		1	Interrupt line to MCANx_INT0 is enabled	
1	EINT1		Enable interrupt line 1.	0
		0	Interrupt line to MCANx_INT1 is disabled	
		1	Interrupt line to MCANx_INT1 is enabled	
31:2	-	-	Reserved.	-

41.8.16 Global filter configuration register

Table 765. Global filter configuration register (GFC, offset 0x080) bit description

Bit	Symbol	Value	Description	Reset value
0	RRFE		Reject remote frames extended.	0
		0	Filter remote frames with 29-bit extended IDs	
		1	Reject all remote frames with 29-bit extended IDs	
1	RRFS		Reject remote frames standard.	0
		0	Filter remote frames with 11-bit standard IDs	
		1	Reject all remote frames with 11-bit standard IDs	

Table 765. Global filter configuration register (GFC, offset 0x080) bit description

Bit	Symbol	Value	Description	Reset value
3:2	ANFE		Accept non-matching frames extended. Defines how receives messages with 29-bit IDs that do not match any element of the filter list are treated.	0
		0x0	Accept in Rx FIFO 0	
		0x1	Accept in Rx FIFO 1	
		0x2	Reject	
		0x3	Reject	
5:4	ANFS		Accept non-matching frames standard. Defines how receives messages with 11-bit IDs that do not match any element of the filter list are treated.	0
		0x0	Accept in Rx FIFO 0	
		0x1	Accept in Rx FIFO 1	
		0x2	Reject	
		0x3	Reject	
31:6	-	-	Reserved.	-

41.8.17 Standard ID filter configuration register

The standard ID filter configuration register controls the filter path for standard messages and settings for the 11-bit standard message ID filter.

Table 766. Standard ID filter configuration register (SIDFC, offset 0x084) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
15:2	FLSSA	Filter list standard start address. Start address of the standard message ID filter list	
23:16	LSS	List size standard. 0 = No standard message ID filter 1 – 128 = Number of standard message ID filter elements >128 = Values of greater than 128 are interpreted as 128	
31:24	-	Reserved.	-

41.8.18 Extended ID filter configuration register

The extended ID filter configuration register controls the filter path for extended messages and settings for the 29-bit extended message ID filter.

Table 767. Extended ID filter configuration register (XIDFC, offset 0x088) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
15:2	FLESA	Filter list extended start address. Start address of the extended message ID filter list.	
23:16	LSE	List size extended. 0 = No extended message ID filter 1 – 64 = Number of extended message ID filter elements >64 = Values of greater than 64 are interpreted as 64	
31:24	-	Reserved.	-

41.8.19 Extended ID AND mask register

The extended ID AND mask register controls the acceptance filtering of extended frames. The value specified in the EIDM bits is ANDed with the message ID of a received frame. The reset value of the EIDM bits is all 1's, which makes the mask not active.

Table 768. Extended ID AND mask register (XIDAM, offset 0x08C) bit description

Bit	Symbol	Description	Reset value
28:0	EIDM	Extended ID mask.	0x1FFF FFFF
31:29	-	Reserved.	-

41.8.20 High priority message status register

The high priority message status register is updated every time a message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Table 769. High priority message status register (HPMS, offset 0x094) bit description

Bit	Symbol	Value	Description	Reset value
5:0	BIDX	-	Buffer index. Index of the Rx FIFO element to which the message was stored. This is only valid when the MSI bits are configured to store the message in either FIFO 0 or FIFO 1 (MSI = 0x2 or 0x3).	0
7:6	MSI		Message storage indicator.	0
		0x0	No FIFO selected	
		0x1	FIFO message lost	
		0x2	Message stored in FIFO 0	
		0x3	Message stored in FIFO 1	
14:8	FIDX	-	Filter index. Index of matching filter element. The range is 1 minus the LSS bits in the SIDFC register resp. LSE bits in the XIDFC registers.	0
15	FLST		Filter list. Indicates the filter list of the matching filter element.	
		0	Standard filter list	
		1	Extended filter list	
31:16	-	-	Reserved.	-

41.8.21 New data 1 register

The new data 1 register holds the new data flags of Rx buffers 0 to 31. The flags are set when the respective Rx buffer has been updated from a received frame. The flags remain set until they are cleared. A flag is cleared by writing a 1 to the corresponding bit position.

Table 770. New data 1 register (NDAT1, offset 0x098) bit description

Bit	Symbol	Description	Reset value
31:0	ND	New data. The most significant bit in this register corresponds to Rx buffer 31 while the least significant bit in this register corresponds to Rx buffer 0.	0

41.8.22 New data 2 register

The new data 2 register holds the new data flags of Rx buffers 32 to 63. The flags are set when the respective Rx buffer has been updated from a received frame. The flags remain set until they are cleared. A flag is cleared by writing a 1 to the corresponding bit position.

Table 771. New data 2 register (NDAT2, offset 0x098) bit description

Bit	Symbol	Description	Reset value
31:0	ND	New data. The most significant bit in this register corresponds to Rx buffer 63 while the least significant bit in this register corresponds to Rx buffer 32.	0

41.8.23 Rx FIFO 0 configuration register

Table 772. Rx FIFO 0 configuration register (RXF0C, offset 0x0A0) bit description

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved.	-
15:2	F0SA	-	Rx FIFO 0 start address. Start address of Rx FIFO 0 in message RAM	0
22:16	F0S	-	Rx FIFO 0 size. 0 = No Rx FIFO 0 1 – 64 = Number of Rx FIFO 0 elements >64 = Values greater than 64 are interpreted as 64 The maximum Rx FIFO 0 elements are the value set in this register minus 1.	
23	-	-	Reserved.	0
30:24	F0WM	-	Rx FIFO 0 watermark. 0 = Watermark interrupt disabled 1 – 64 = Level for RF0W interrupt flag in the IR register >64 = Watermark interrupt disabled	
31	F0OM		FIFO 0 operation mode. The FIFO can be operated in block or overwrite mode.	0
		0	FIFO 0 blocking mode	
		1	FIFO 0 overwrite mode	

41.8.24 Rx FIFO 0 status register

Table 773. Rx FIFO 0 status register (RXF0S, offset 0x0A4) bit description

Bit	Symbol	Value	Description	Reset value
6:0	F0FL	-	Rx FIFO 0 fill level. Number of elements stored in Rx FIFO0, range 0 to 64.	0
7	-	-	Reserved.	-
13:8	F0GI	-	Rx FIFO 0 get index. Rx FIFO 0 read index pointer, range 0 to 63.	0
15:14	-	-	Reserved.	-
21:16	F0PI	-	Rx FIFO 0 put index. Rx FIFO 0 write index pointer, range 0 to 63.	0
23:22	-	-	Reserved.	-
24	F0F		Rx FIFO 0 full.	0
		0	Rx FIFO 0 not full	
		1	Rx FIFO 0 full.	
25	RF0L	-	Rx FIFO 0 message lost. This bit is a copy of the RF0L interrupt flag in the IR register. **Remark:** Overwriting the oldest message when the F0OM bit in the RXF0C register is set to 1 will not set this flag.	0
31:26	-	-	Reserved.	-

41.8.25 Rx FIFO 0 acknowledge register

After a message or sequence of messages have been read from Rx FIFO 0, the buffer index of the last element read from the Rx FIFO 0 must be written to the F0AI bits. This will set the Rx FIFO 0 get index bits to the Rx FIFO 0 acknowledge index bit value + 1 and update the FIFO 0 fill level bits.

Table 774. Rx FIFO 0 acknowledge register (RXF0A, offset 0x0A8) bit description

Bit	Symbol	Description	Reset value
5:0	F0AI	Rx FIFO 0 acknowledge index.	0
31:6	-	Reserved.	-

41.8.26 Rx buffer configuration register

The x buffer configuration register is used to configure the start address of the Rx buffers section in the message RAM.

Table 775. Rx buffer configuration register (RXBC, offset 0x0AC) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
15:2	RBSA	Rx buffer start address.	0
31:16	-	Reserved.	-

41.8.27 Rx FIFO 1 configuration register

Table 776. Rx FIFO 1 configuration register (RXF1C, offset 0x0B0) bit description

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved.	-
15:2	F1SA	-	Rx FIFO 1 start address. Rx FIFO 1 start address. Start address of Rx FIFO 1 in message RAM.	0
22:16	F1S	-	Rx FIFO 1 size. 0 = No Rx FIFO 1 1 – 64 = Number of Rx FIFO 1 elements >64 = Values greater than 64 are interpreted as 64 The maximum Rx FIFO 1 elements are the value set in this register minus 1.	-
23	-	-	Reserved.	-
30:24	F1WM	-	Rx FIFO 1 watermark. 0 = Watermark interrupt disable 1 – 64 = Level for RF1W interrupt flag in the IR register. >64 = Watermark interrupt disabled	0
31	F1OM		FIFO 1 operation mode. The FIFO can be operated in block or overwrite mode.	0
		0	FIFO 1 blocking mode	
		1	FIFO 1 overwrite mode	

41.8.28 Rx FIFO 1 status register

Table 777. Rx FIFO 1 status register (RXF1S, offset 0x0B4) bit description

Bit	Symbol	Value	Description	Reset value
6:0	F1FL	-	Rx FIFO 1 fill level. Number of elements stored in Rx FIFO1, range 0 to 64.	0
7	-	-	Reserved.	-
13:8	F1GI	-	Rx FIFO 1 get index. Rx FIFO 1 read index pointer, range 0 to 63.	0
15:14	-	-	Reserved.	-
21:16	F1PI	-	Rx FIFO 1 put index. Rx FIFO 1 write index pointer, range 0 to 63.	-
23:22	-	-	Reserved.	-
24	F1F		Rx FIFO 1 full.	0
		0	Rx FIFO 1 not full	
		1	Rx FIFO 1 full.	
25	RF1L		Rx FIFO 1 message lost. This bit is a copy of the RF1L interrupt flag in the IR register. **Remark:** overwriting the oldest message when the F1OM bit from the RXF1C register is set to 1 will not set this flag.	
		0	No Rx FIFO 1 message lost.	
		1	Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero.	
31:26	-	-	Reserved.	-

41.8.29 Rx FIFO 1 acknowledge register

After a message or sequence of messages have been read from Rx FIFO 1, the buffer index of the last element read from the Rx FIFO 1 has to be written to the F1AI bits. This will set the Rx FIFO 1 get index bits to the Rx FIFO 1 acknowledge index bit value + 1 and update the FIFO 1 fill level bits.

Table 778. Rx FIFO 1 acknowledge register (RXF1A, offset 0x0B8) bit description

Bit	Symbol	Description	Reset value
5:0	F1AI	Rx FIFO 1 acknowledge index.	0
31:6	-	Reserved.	-

41.8.30 Rx buffer and FIFO element size configuration register

The Rx buffer and FIFO element size configuration register configures the number of data bytes belonging to the Rx buffer and Rx FIFO element. Data field sizes that are greater than 8 bytes are intended for CAN FD operation only.

Table 779. Rx buffer and FIFO element size configuration register (RXESC, offset 0x0BC) bit description

Bit	Symbol	Value	Description	Reset value
2:0	F0DS		Rx FIFO 0 data field size.	0
		0x0	8 byte data field	
		0x1	12 byte data field	
		0x2	16 byte data field	
		0x3	20 byte data field	
		0x4	24 byte data field	
		0x5	32 byte data field	
		0x6	48 byte data field	
		0x7	64 byte data field	
3	-	-	Reserved.	-
6:4	F1DS		Rx FIFO 1 data field size.	0
		0x0	8 byte data field	
		0x1	12 byte data field	
		0x2	16 byte data field	
		0x3	20 byte data field	
		0x4	24 byte data field	
		0x5	32 byte data field	
		0x6	48 byte data field	
		0x7	64 byte data field	
7	-	-	Reserved.	-
10:8	RBDS		Rx buffer data field size.	0
		0x0	8 byte data field	
		0x1	12 byte data field	
		0x2	16 byte data field	
		0x3	20 byte data field	
		0x4	24 byte data field	
		0x5	32 byte data field	
		0x6	48 byte data field	
		0x7	64 byte data field	
31:11	-	-	Reserved.	-

Remark: In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx buffer or Rx FIFO, only the number of bytes as configured by RXESC register are stored to the Rx buffer resp. Rx FIFO element. The data field of the rest of the frame is ignored.

41.8.31 Tx buffer configuration register

Table 780. Tx buffer configuration register (TXBC, offset 0x0C0) bit description

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved.	-
15:2	TBSA	-	Tx buffers start address. Start address of Tx buffers in message RAM.	0
21:16	NDTB	-	Number of dedicated transmit buffers. 0 = No dedicated Tx buffers 1 – 32 = Number of dedicated Tx buffers >32 = Values greater than 32 are interpreted as 32	-
23:22	-	-	Reserved.	-
29:24	TFQS	-	Transmit FIFO/queue size. 0 = No tx FIFO/Queue 1 – 32 = Number of Tx buffers used for Tx FIFO/Queue >32 = Values greater than 32 are interpreted as 32	0
30	TFQM		Tx FIFO/queue mode.	0
		0	Tx FIFO operation	
		1	Tx queue operation	
31	-	-	Reserved.	-

Remark: The sum of TFQS and NDTB bits may not be greater than 32. There is no check for erroneous configurations. The Tx buffers section in the Message RAM starts with the dedicated Tx buffers.

41.8.32 Tx FIFO/queue status register

The Tx FIFO/queue status is related to the pending Tx requests listed in the TXBRP register. Therefore the effect of add/cancellation requests may be delayed due to a running Tx scan.

Table 781. Tx FIFO/queue status register (TXFQS, offset 0x0C4) bit description

Bit	Symbol	Value	Description	Reset value
7:0	-	-	Reserved.	-
12:8	TFGI	-	Tx FIFO get index. Tx FIFO read index pointer, range 0 to 31.	-
15:13	-	-	Reserved.	-
20:16	TFQPI	-	Tx FIFO/queue put index. Tx FIFO/queue write index pointer, range 0 to 31.	0
21	TFQF		Tx FIFO/queue full.	0
		0	Tx FIFO/queue not full	
		1	Tx FIFO/queue full	
31:22	-	-	Reserved.	-

Remark: In case of mixed configurations where dedicated Tx buffers are combined with a Tx FIFO or a Tx queue, the put and get indices indicate the number of the Tx buffer starting with the first dedicated Tx buffers.

For example, a configuration of 12 dedicated Tx buffers and a Tx FIFO of 20 buffers a put index of 15 points to the fourth buffer of the Tx FIFO.

41.8.33 Tx buffer element size configuration register

Table 782. Tx buffer element size configuration register (TXESC, offset 0x0C8) bit description

Bit	Symbol	Value	Description	Reset value
2:0	TBDS	-	Tx buffer data field size.	0
		0x0	8 byte data field	
		0x1	12 byte data field	
		0x2	16 byte data field	
		0x3	20 byte data field	
		0x4	24 byte data field	
		0x5	32 byte data field	
		0x6	48 byte data field	
		0x7	64 byte data field	
31:3	-	-	Reserved.	-

Remark: In case the data length code of a Tx buffer element is configured to a value higher than the TBDS bits, the bytes not defined by the Tx buffer are transmitted as “0xCC” as padding.

41.8.34 Tx buffer request pending register

Each Tx buffer has its own transmission request pending bit. The bits are set in the TXBAR register. The bits are reset after a requested transmission has completed or has been cancelled via the TXBCR register.

The TXBRP bits are set only for those Tx buffers configured in the TXBC register. After a TXBRP bit has been set, a Tx scan is started to check the pending Tx request with the highest priority.

A cancellation request resets the corresponding transmission request pending bit in the TXBRP register. In case a transmission has already been started when a cancellation is requested, it is done at the end of the transmission, regardless of the transmission success. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset.

After a cancellation has been requested, a finished cancellation is signaled using the TXBCF register:

- After a successful transmission together with the corresponding TXBTO bit.
- When a transmission has not yet been started at the point of cancellation.
- When the transmission has been aborted due to lost arbitration.
- When an error occurred during frame transmission.

In DAR mode, all transmissions are automatically cancelled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions.

Table 783. Tx buffer request pending register (TXBRP, offset 0x0CC) bit description

Bit	Symbol	Value	Description	Reset value
31:0	TRP		Transmission request pending.	0
		0	No transmission request pending	
		1	Transmission request pending	

Remark: TXBRP bits, which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is cancelled immediately and the corresponding TXBRP bit is reset.

41.8.35 Tx buffer add request register

Each Tx buffer has its own add request bit. The number of Tx buffers is configured in the TXBC register. Writing a 1 will set the corresponding add request bit while writing a 0 has no impact. This allows for setting transmission requests for multiple Tx buffers with one write. The TXBAR bits are only set for those Tx buffers configured in the TXBC register. When no Tx scan is running, the bits are reset immediately, otherwise the bits remain set until the Tx scan process is completed.

Table 784. Tx buffer add request register (TXBAR, offset 0x0D0) bit description

Bit	Symbol	Value	Description	Reset value
31:0	AR		Add request.	0
		0	No transmission request added	
		1	Transmission request added	

Remark: If an add request is applied for a Tx buffer with pending transmission request, the add request is ignored.

41.8.36 Tx buffer cancellation request register

Each Tx buffer has its own add request bit. The number of Tx buffers is configured in the TXBC register. Writing a 1 will set the corresponding add request bit while writing a 0 has no impact. This allows for setting transmission requests for multiple Tx buffers with one write. The TXBCR bits are only set for those Tx buffers configured in the TXBC register. The bits remain set until the corresponding bit in the TXBRP register is reset.

Table 785. Tx buffer cancellation request register (TXBCR, offset 0x0D4) bit description

Bit	Symbol	Value	Description	Reset value
31:0	CR		Cancellation request.	0
		0	No cancellation pending	
		1	Cancellation pending	

41.8.37 Tx buffer transmission occurred register

Each Tx buffer has its own transmission occurred bit. The bits are set when the corresponding bits in the TXBRP register is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit in the TXBAR register.

Table 786. Tx buffer transmission occurred register (TXBTO, offset 0x0D8) bit description

Bit	Symbol	Value	Description	Reset value
31:0	TO		Transmission occurred.	0
		0	No transmission occurred	
		1	Transmission occurred	

41.8.38 Tx buffer cancellation finished register

Each Tx buffer has its own cancellation finished bit. The bits are set when the corresponding bits in the TXBRP register is cleared after a cancellation was requested in the TXBCR register. In case the corresponding bits in the TXBRP register was not set at the point of cancellation, the cancellation finished bits in this register will be set immediately. The bits are reset when a new transmit cancellation is requested by writing a 1 to the corresponding bit in the TXBAR register.

Table 787. Tx buffer cancellation finished register (TXBCF, offset 0x0DC) bit description

Bit	Symbol	Value	Description	Reset value
31:0	TO		Cancellation finished.	0
		0	No transmit buffer cancellation	
		1	Transmit buffer cancellation finished	

41.8.39 Tx buffer transmission interrupt enable register

Table 788. Tx buffer transmission interrupt enable register (TXBTIE, offset 0x0E0) bit description

Bit	Symbol	Value	Description	Reset value
31:0	TIE		Transmission interrupt enable. Each Tx buffer has its own transmission interrupt enable bit.	0
		0	Transmission interrupt disabled	
		1	Transmission interrupt enabled.	

41.8.40 Tx buffer cancellation finished interrupt enable register

Table 789. Tx buffer cancellation finished interrupt enable register (TXBCIE, offset 0x0E4) bit description

Bit	Symbol	Value	Description	Reset value
31:0	CFIE		Cancellation finished interrupt enable. Each Tx buffer has its own transmission interrupt enable bit.	0
		0	Cancellation finished interrupt disabled	
		1	Cancellation finished interrupt enabled	

41.8.41 Tx event FIFO configuration register

Table 790. Tx event FIFO configuration register (TXEFC, offset 0x0F0) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
15:2	EFSA	Event FIFO start address. Start address of the Tx event FIFO in message RAM.	0
21:16	EFS	Event FIFO size. 0 = Tx event FIFO disabled 1 – 32 = Number of Tx event FIFO elements >32 = Values greater than 32 are interpreted as 32 The maximum Tx FIFO elements are the value set in this register minus 1.	0
23:22	-	Reserved.	-
29:24	EFWM	Event FIFO watermark. 0 = Watermark interrupt disabled 1 – 32 = Level for TEFW interrupt flag in the IR register >32 = Watermark interrupt disabled	0
31:30	-	Reserved.	-

41.8.42 Tx event FIFO status register

Table 791. Tx event FIFO status register (TXEFS, offset 0x0F4) bit description

Bit	Symbol	Value	Description	Reset value
5:0	EFFL	-	Event FIFO fill level. Number of elements stored in Tx event FIFO, range 0 to 32.	0
7:6	-	-	Reserved.	-
12:8	EFGI	-	Event FIFO get index. Tx event FIFO read index pointer, range 0 to 31.	0
15:13	-	-	Reserved.	-
21:16	EFPI	-	Event FIFO put index. Tx event FIFO write index pointer, range 0 to 31.	0
23:22	-	-	Reserved.	-
24	EFF		Event FIFO full.	0
		0	Tx event FIFO not full	
		1	Tx event FIFO full	
25	TEFL		Tx event FIFO element lost. This bit is a copy of the TEFL interrupt flag in the IR register. When the TEFL bit is reset, this bit is also reset.	0
		0	No Tx event FIFO element lost.	
		1	Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size zero.	
31:26	-	-	Reserved.	-

41.8.43 Tx event FIFO acknowledge register

After an element or sequence of elements have been read from the Tx event FIFO, the index of the last element read has to be written to the event FIFO acknowledge index bits in the EFAI bits. This will set the Tx event FIFO get index bits to the value stored in the EFAI +1 and update the event FIFO fill level.

Table 792. Tx event FIFO acknowledge register (TXEFA, offset 0x0F8) bit description

Bit	Symbol	Description	Reset value
4:0	EFAI	Event FIFO acknowledge index.	0
31:5	-	Reserved.	-

41.8.44 Message RAM base address register

Table 793. Message RAM base address register (MRBA, offset 0x200) bit description

Bit	Symbol	Description	Reset value
15:0	-	Reserved. Read value is undefined, only 0 should be written.	0
31:16	BA	Base address for the message RAM in the chip memory map. Bits 0 to 15 are reserved and 0x0000 should be written to these bits.	0

41.8.45 External timestamp counter configuration register

Table 794. External timestamp counter configuration register (ETSCC, offset 0x400) bit description

Bit	Symbol	Value	Description	Reset value
10:0	ETCP	-	External timestamp prescaler value. The CPUCLK is divided down by the prescaler value plus 1 to clock the external timestamp counter.	0
30:11	-	-	Reserved.	-
31	ETCE		External timestamp counter enable.	0
		0	External timestamp counter is disabled	
		1	External timestamp counter is enabled	

41.8.46 External timestamp counter value register

Table 795. External timestamp counter value register (ETSCV, offset 0x600) bit description

Bit	Symbol	Description	Reset value
15:0	ETSC	External timestamp counter. Read to return current counter value. Write to initialize the counter with the specified 16-bit value.	0
31:16	-	Reserved.	-

41.9 Rx buffer and FIFO element

Up to 64 Rx buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of an Rx buffer / FIFO element is shown in Figure 132. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via the RXESC register.

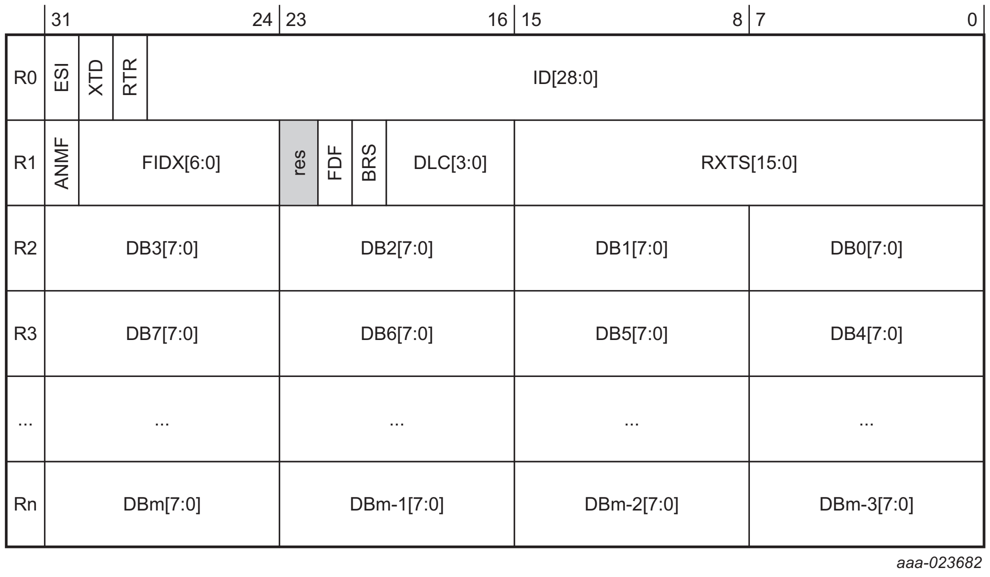


Fig 132. Rx buffer and FIFO element

Table 796. R0 bit description

Bit	Symbol	Value	Description
28:0	ID	-	Identifier. Standard or extended identifier depending on the XTD bit. A standard identifier is expected to be stored into ID bits 28:18.
29	RTR		Remote transmission request. This bit is set depending on whether the received frame is a data frame or a remote frame. **Remark:** There are no remote frames in CAN FD format.
		0	Received frame is a data frame
		1	Received frame is a remote frame
30	XTD		Extended identifier.
		0	11-bit standard identifier
		1	29-bit extended identifier
31	ESI		Error state identifier.
		0	Transmitting node is error active
		1	Transmitting node is error passive

Table 797. R1 bit description

Bit	Symbol	Value	Description
15:0	RXTS	-	Rx timestamp. Timestamp counter value captures on start of frame reception. Resolution depending on configuration of timestamp counter prescaler bit in the TSCC register.
19:16	DLC	-	Data length code. 0 – 8 = CAN + CAN FD: received frame has 0 - 8 data bytes 9 – 15 = CAN: received frame has 8 data bytes 9 – 15 = CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
20	BRS		Bit rate switch.
		0	Frame received without bit rate switching
		1	Frame received with bit rate switching
21	FDF		FD format.
		0	Standard frame format.
		1	CAN FD frame format (new DLC-coding and CRC).
31	ESI		Error state identifier
		0	Transmitting node is error active
		1	Transmitting node is error passive
23:22	-	-	Reserved.
30:24	FIDX	-	Filter index. 0 – 127 = Index of matching Rx acceptance filter element (invalid if the ANMF bit is set) Range is 0 to one minus the LSS bits in the SIDFC register resp. the LSE bit in the XIDFC register
31	ANMF		Accepted non-matching frame. Acceptance of non-matching frames may be enabled via the ANFS and ANFE bits in the GFC register.
		0	Received frame matching filter index FIDX
		1	Received frame did not match any Rx filter element

Table 798. R2 bit description

Bit	Symbol	Description
7:0	DB	Data byte.
15:8	DB	Data byte.
23:16	DB	Data byte.
31:24	DB	Data byte.

Remark: Depending on the configuration of the element size in the RXESC register, between two and sixteen 32-bit words are used for storage of a CAN message’s data field.

41.10 Tx buffer element

The Tx buffers section can be configured to hold dedicated Tx buffers as well as a Tx FIFO / Tx Queue. In case that the Tx buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx buffers start at the beginning of the Tx buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx buffers and Tx FIFO / Tx Queue by evaluating the TFQS and NDTB bits in the TXBC register. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via the TXESC register.

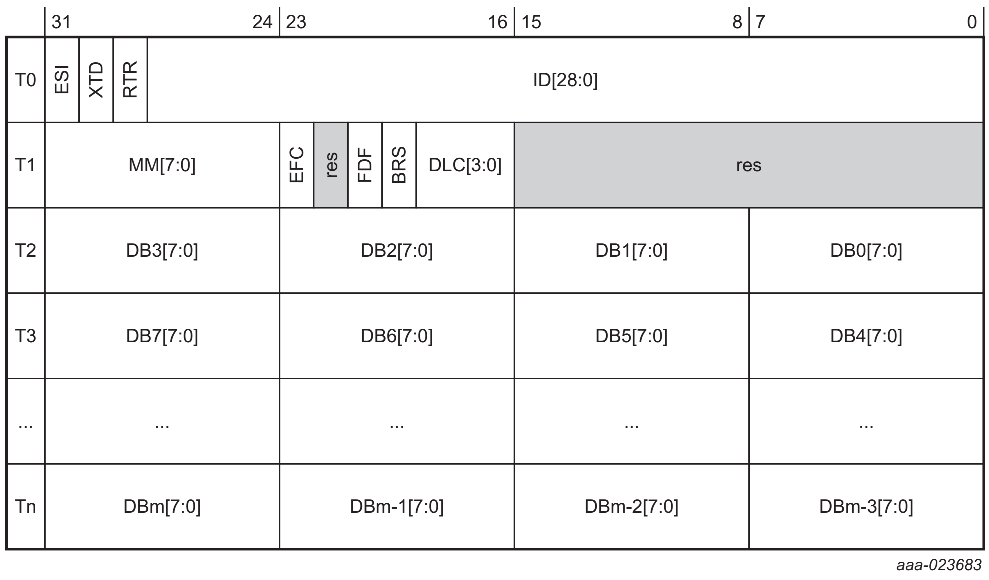


Fig 133. Tx buffer element

Table 799. T0 bit description

Bit	Symbol	Value	Description
28:0	ID	-	Identifier. Standard or extended identifier depending on the XTD bit. A standard identifier is expected to be stored into ID bits 28:18.
29	RTR		Remote transmission request. **Remark:** When this bit is set, the MCAN transmits a remote frame according to ISO11898-1, even if the FDOE bit of the CCCR register enables the transmission in CAN FD format.
		0	Transmit data frame
		1	Transmit remote frame
30	XTD		Extended identifier.
		0	11-bit standard identifier
		1	29-bit extended identifier

Table 799. T0 bit description

Bit	Symbol	Value	Description
31	ESI		Error state identifier. **Remark:** The ESI bit of the transmit buffer is OR’d with the passive flag to decide the value of the ESI bit in the transmitted FD frame. Per the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit ESI bit recessive.
		0	ESI bit in CAN FD format depends only on error passive flag
		1	ESI bit in CAN FD format transmitted recessive

Table 800. T1 bit description

Bit	Symbol	Value	Description
15:0	-	-	Reserved.
19:16	DLC	-	Data length code. 0 – 8 = CAN + CAN FD: transmit frame has 0 - 8 data bytes; 9 – 15 = CAN: transmit frame has 8 data bytes; 9 – 15 = CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes
20	BRS		Bit rate switch.
		0	CAN FD frames transmitted without bit rate switching
		1	CAN FD frames transmitted with bit rate switching
21	FDF		FD format.
		0	Frame transmitted in classic CAN format
		1	Frame transmitted in CAN FD format
22	-		Reserved.
23	EFC		Event FIFO control.
		0	Do not store Tx events
		1	Store Tx events
31:24	MM	-	Message Marker. Written during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.

Table 801. R2 to Rn bit description

Bit	Symbol	Description
7:0	DB	Data byte.
15:8	DB	Data byte.
23:16	DB	Data byte.
31:24	DB	Data byte.

Remark: Depending on the configuration of the element size in the RXESC register, between two and sixteen 32-bit words are used to store the data field of a CAN message.

41.11 Tx event FIFO element

Each element stores information about transmitted messages that can be used to get information of the order of transmitted messages. Status information about the Tx event FIFO can be obtained from the TXEFS register.

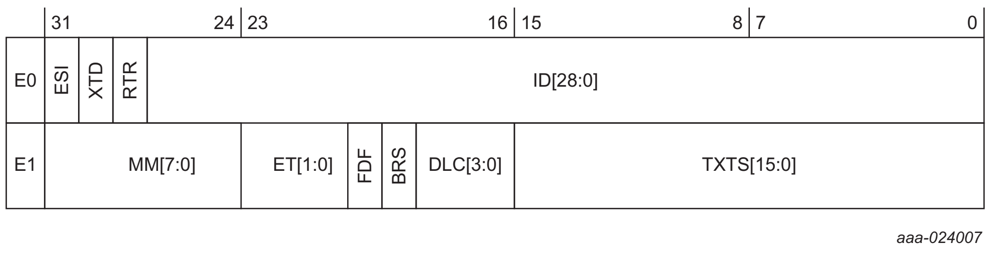


Fig 134. Tx event FIFO element

Table 802. E0 bit description

Bit	Symbol	Value	Description
28:0	ID	-	Identifier. Standard or extended identifier depending on the XTD bit. A standard identifier is expected to be stored into ID bits 28:18.
29	RTR		Remote transmission request.
		0	Data frame transmitted
		1	Remote frame transmitted
30	XTD		Extended identifier.
		0	11-bit standard identifier
		1	29-bit extended identifier
31	ESI		Error state identifier.
		0	Transmitted node is error active
		1	Transmitted node is error passive

Table 803. E1 bit description

Bit	Symbol	Value	Description
15:0	TXTS	-	Tx timestamp. Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the TCP bit in the TSCC register.
19:16	DLC	-	Data length code. 0 – 8 = CAN + CAN FD: frame with 0 - 8 data bytes transmitted 9 – 15 = CAN: frame with 8 data bytes transmitted 9 – 15 = CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted
20	BRS		Bit rate switch.
		0	Frames transmitted without bit rate switching
		1	Frames transmitted with bit rate switching
21	FDF		FD Format.
		0	Standard frame format
		1	CAN FD frame format (new DLC-coding and CRC)

Table 803. E1 bit description *…continued*

Bit	Symbol	Value	Description
23:22	ET		Event type.
		0x0	Reserved
		0x1	Tx event
		0x2	Transmission in spite of cancellation (always set for transmission in DAR mode)
		0x3	Reserved
31:24	MM	-	Message Marker. Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.

41.12 Standard message ID filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is equal to the filter list standard start address bit field in the SIDFC register plus the index of the filter element.

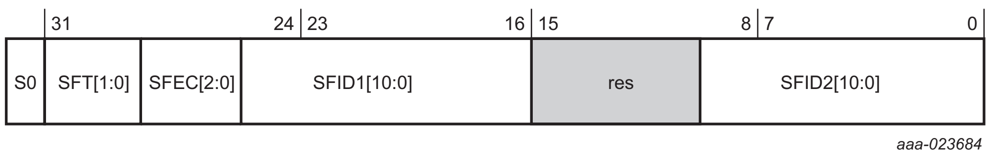


Fig 135. Standard message ID filter element

Table 804. S0 bit description

Bit	Symbol	Value	Description
10:0	SFID2	-	Standard filter ID 2. This bit field has a different meaning depending on the configuration of the SFEC register. 1. SFEC = 0x1 – 0x6, this bit field becomes the second ID of standard ID filter element. 2. SFEC = 0x7, this bit field is used to filter for Rx buffers or for debug messages. Bits 10:9 decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. 0x0 = Store message into an Rx buffer 0x1 = Debug message A 0x2 = Debug message B 0x3 = Debug message C Bits 8:6 are used to control the filter event pins at the extension interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one MCAN function clock period in case the filter matches. Bits 5:0 define the offset to the Rx buffer start address bit field in the RXBC register for storage of a matching message.
15:11	-	-	Reserved.
26:16	SFID1	-	Standard filter ID 1. First ID of standard ID filter element. When filtering for Rx buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.

Table 804. S0 bit description

Bit	Symbol	Value	Description
29:27	SFEC		Standard filter element configuration. All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If this bit field is set to 0x4, 0x5, or 0x6, a match sets the HPM interrupt flag in the IR register, if enabled, an interrupt is generated. In this case, the HPMS register is updated with the status of the priority match.
		0x0	Disable filter element
		0x1	Store in Rx FIFO 0 if filter matches
		0x2	Store in Rx FIFO 1 if filter matches
		0x3	Reject ID if filter matches
		0x4	Set priority if filter matches
		0x5	Set priority and store in FIFO 0 if filter matches
		0x6	Set priority and store in FIFO 1 if filter matches
		0x7	Store into Rx buffer or as debug message, configuration of the SFT bit field is ignored
31:30	SFT		Standard filter type.
		0x0	Range filter from SFID1 to SFID2 (SFID2 ≥ SFID1)
		0x1	Dual ID filter for SFID1 or SFID2
		0x2	Classic filter: SFID1 = filter, SFID2 = mask
		0x3	Filter element disabled

41.13 Extended message ID filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an extended message ID filter element, its address is equal to the filter list extended start address bit field in the XIDFC register plus two times the index of the filter element.

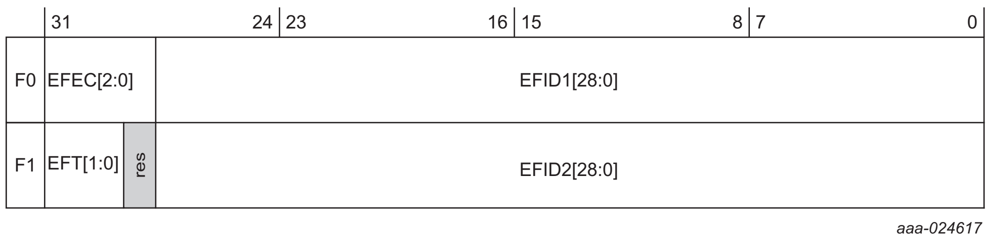


Fig 136. Extended message ID filter element

Table 805. F0 bit description

Bit	Symbol	Value	Description
28:0	EFID1	-	Extended filter ID 1. First ID of extended ID filter element. When filtering for Rx buffers or for debug messages, this field defines the ID of an extended message to be stored. The received identifiers must match exactly when masked with the XIDAM register.
31:29	EFEC		Extended filter element configuration. All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If this bit field is equal to 0x4, 0x5, or 0x6, a match sets the HPM interrupt flag in the IR register and, if enabled, an interrupt is generated. In this case, the HPMS register is updated with the status of the priority match.
		0x0	Disable filter element
		0x1	Store in Rx FIFO 0 if filter matches
		0x2	Store in Rx FIFO 1 if filter matches
		0x3	Reject ID if filter matches
		0x4	Set priority if filter matches
		0x5	Set priority and store in FIFO 0 if filter matches
		0x6	Set priority and store in FIFO 1 if filter matches
		0x7	Store into Rx buffer or as debug message, configuration of the EFT bit field is ignored

Table 806. F1 bit description

Bit	Symbol	Value	Description
28:0	EFID2	-	Extended filter ID 2. This bit field has a different meaning depending on the configuration of the EFEC bit field. 1. EFEC = 0x1 – 0x6, this bit field becomes the second ID of extended ID filter element. 2. EFEC = 0x7, this bit field is used to filter for Rx buffers or debug messages. Bits 10:9 decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. 0x0 = Store message into an Rx buffer 0x1 = Debug message A 0x2 = Debug message B 0x3 = Debug message C Bits 8:6 is used to control the filter event pins at the extension interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one MCAN function clock period in case the filter matches. Bits 5:0 defines the offset to the Rx buffer start address bit field in the RXBC register for storage of a matching message.
29	-	-	Reserved.
31:30	EFT		Extended filter type.
		0x0	Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1)
		0x1	Dual ID filter for EFID1 or EFID2
		0x2	Classic filter: EFID1 = filter, EFID2 = mask
		0x3	Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1), the mask in the XIDAM register is not applied

41.14 Functional description

41.14.1 Operating modes

41.14.1.1 Software initialization

Software initialization is started by setting the INIT bit in the CCCR register, either by software or by a hardware reset, when an uncorrected bit error was detected in the message RAM, or by going Bus_Off. While the INIT bit is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output m_can_tx is recessive. The counters of the error management logic are unchanged. Setting the INIT bit does not change any configuration register. Resetting the INIT bit finishes the software initialization. Afterwards, the bit stream processor synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits, indicating the bus is idle, before it can take part in bus activities and start the message transfer.

Access to the MCAN configuration registers is only enabled when both the INIT and CCE bits are set in the CCCR register.

The CCE bit can only be set or reset while the INIT bit is set. The CCE bit is automatically reset when the INIT bit is reset.

The following registers are reset when the CCE bit is set:

- HPMS – High priority message status.
- RXF0S – Rx FIFO 0 status.
- RXF1S – Rx FIFO 1 status.
- TXFQS – Tx FIFO/queue status.
- TXBTO – Tx buffer transmission occurred.
- TXBRP – Tx buffer request pending.
- TXBCF – Tx buffer cancellation finished.
- TXEFS – Tx event FIFO status.

The timeout counter value in the TOCV register is preset to the value configured by the TOP bits in the TOCC register, while the CCE bit is set in the CCCR register. In addition, the state machines of the Tx and Rx handlers are held in an idle state while the CCE bit is set.

The following registers are only writable while the CCE bit is cleared:

- TXBAR – Tx buffer add request.
- TXBCR – Tx buffer cancellation request.

The TEST and MON bits in the CCCR register can only be set while the INIT and CCE bits are set. Both bits may be reset at any time. The DAR bit can only be set or reset while the INIT and CCE bit are set.

41.14.1.2 Normal operation

When the MCAN is initialized and the INIT bit in the CCCR register is cleared, the MCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into a dedicated Rx buffer or into Rx FIFO 0 or Rx FIFO 1.

To transmit messages, dedicated Tx buffers and/or a Tx FIFO or a Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

41.14.1.3 CAN FD operation

There are two variants in the CAN FD frame transmission. The first is the CAN FD frame without bit rate switching. The second variant is the CAN FD frame where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF = recessive signifies a CAN FD frame, FDF = dominant signifies a classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = dominant and BRS = recessive. The coding of res = recessive is reserved for future expansion of the protocol. In case the MCAN receives a frame with FDF = recessive and res = recessive, it will signal a Protocol Exception Event by setting bit PSR.PXE. When Protocol Exception Handling is enabled in the CCCR register, this causes the operation state to change from Receiver at the next sample point, see the ACT bits in the PSR register. In case Protocol Exception Handling is disabled in the CCCR register, the MCAN will treat a recessive res bit as a form error and will respond with an error frame.

With FDOE bit cleared in the CCCR register, the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format. With FDOE bit set and the BRSE bit cleared in the CCCR register, only bit FDF of a Tx buffer element is evaluated. With FDOE and BRSE bit set in the CCCR register, transmission of CAN FD frames with bit rate switching is enabled. All Tx buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significantly higher than in the CAN FD arbitration phase. In this case, disable the CAN FD bit rate switching option for transmissions.
- During system startup, all nodes are transmitting classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking have to be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in silent mode until programming has completed. Afterwards, all nodes switch back to classic CAN communication.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in the standard CAN. Table 807 shows the DLC codes 9 to 15. In the standard CAN, codes 9 to 15 code a data field of 8 bytes.

Table 807. DLC coding in CAN FD

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD frames, the bit timing will be switched inside the frame, after the Bit Rate Switch (BRS) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the NBTP register. In the following CAN FD data phase, the data phase bit timing is used as defined by the DBTP register. The bit timing is switched back from the data phase timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency. For example, if the CAN clock frequency is 20 MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD and CAN FD with bit rate switching, the value of the Error Status Indicator (ESI) bit is determined by the error state of the transmitter at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, otherwise it is transmitted dominant.

41.14.1.4 Restricted operation mode

Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The REC and TEC error counters in the ECR register are frozen while the CEL bit is set. The MCAN can be put into restricted operation mode by setting the ASM bit in the CCCR register. The bit can only be set when CCE and INIT bits are set in the CCCR register. The bit can be cleared at any time.

Restricted operation mode is automatically entered when the Tx Handler was not able to read data from the message RAM in time. To leave restricted operation mode, the ASM bit in the CCCR register must be cleared.

The restricted operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the restricted operation mode after it has received a valid frame.

If the MCAN is connected to the clock calibration on the CAN unit, the ASM bit in the CCCR register is controlled by input m_can_cok. In case MCAN_COK switches to 0, the ASM bit is set. When the m_can_cok switches back to 1, the ASM bit returns to the previously written value. When there is no clock calibration on the CAN unit's connected input, m_can_cok is hardwired to 1.

Remark: The restricted operation mode must not be combined with the loop back mode, whether internal or external.

41.14.1.5 Bus monitoring mode

The MCAN is set in bus monitoring mode by setting the MON bit in the CCCR registers. In bus monitoring mode (see ISO11898-1, 10.12 Bus monitoring), the MCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the MCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the MCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In bus monitoring mode, the TXBRP register is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. Figure 137 shows the connection of the m_can_tx and m_can_rx signals to the MCAN in bus monitoring mode.

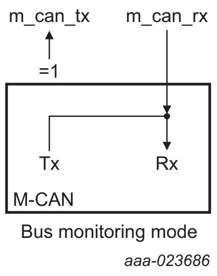


Fig 137. Pin control in bus monitoring mode

41.14.1.6 MCAN power down mode (sleep mode)

The MCAN can be set into power down mode controlled by the CSR bit in the CCCR register.

When all pending transmission requests have completed, the M_CAN waits until bus idle state is detected. Then the MCAN sets the INIT bit in the CCCR register to 1 to prevent any more CAN transfers. Now the MCAN acknowledges that it is ready for power down by setting the CSA bit to 1. In this state, before the clocks are switched off, further register accesses can be made. A write access to INIT bit will have no effect.

To leave the MCAN power down mode, the application has to turn on the module clocks before resetting the CSR bit in the CCCR register. The MCAN acknowledges this by resetting the CSA bit. Afterwards, the application can restart CAN communication by clearing the INIT bit.

41.14.1.7 Test modes

To enable write access to the test register, the TEST bit in the CCCR register must be set. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin CAN1_TD by programming the TX bit field in the TEST register. Additionally to its default function – the serial data output – it can drive the CAN sample point signal to monitor the bit timing f the MCAN and it can drive constant dominant or recessive values. The actual value at pin CAN1_RD can be read from the RX bit field in the TEST register. Both functions can be used to check the physical layer of the CAN bus.

The synchronization mechanism in the CAN IP may cause a delay of several clock periods between writing to the TX bit field until the new configuration is visible at output pin m_can_tx. This applies also when reading input pin CAN1_RD via the RX bit field.

Remark: Test modes should be used for production tests or self-tests only. The software control for the m_can_tx interferes with all CAN protocol functions. It is not recommended to use test modes for applications.

41.14.1.7.1 External loop back mode

The MCAN can be set in external loop back mode by setting the LBCK bit in the TEST register. In loop back mode, the MCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into an Rx buffer or an Rx FIFO. Figure 12 shows the connection of signals m_can_tx and CAN1_RD to the MCAN in external loop back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the MCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop back mode. In this mode the MCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN1_RD input pin is disregarded by the MCAN. The transmitted messages can be monitored at the m_can_tx pin.

41.14.1.7.2 Internal loop back mode

Internal loop back mode is entered by setting the LBCK bit in the TEST register and the MON bit in the CCCR register. This mode can be used for a “Hot Selftest”, meaning, the MCAN can be tested without affecting a running CAN system connected to the pins m_can_tx and CAN1_RD. In this mode pin CAN1_RD is disconnected from the MCAN and pin m_can_tx is held recessive. Figure 138 shows the connection of m_can_tx and CAN1_RD to the MCAN in case of internal loop back mode.

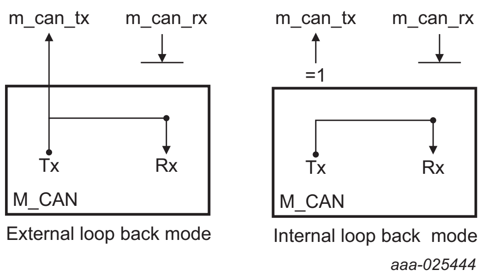


Fig 138. Pin control in loop back modes

41.14.2 Transmitter delay compensation

During the data phase of a CAN FD transmission, only one node transmits while all others are receivers. The length of the bus line has no impact. When transmitting with the CANx_TD pin, the M_CAN receives the transmitted data from its local CAN transceiver via the CANx_RD pin. The received data is delayed by the transmitter delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transmitter delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transmitter delay.

41.14.2.1 Description

The protocol unit of the MCAN has implemented a delay compensation mechanism to compensate the transmitter delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting the TDC bit in the DBTP register.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the MCAN’s transmit output through the transceiver to the receive input plus the transmitter delay compensation offset as configured by TDCO bit field in the TDCR register. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example, half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of MCAN clock periods.

The TDCV bit field in the PSR shows the actual transmitter delay compensation value. This bit field is cleared when INIT bit in the CCCR register is set and is updated at each transmission of an FD frame while TDC bit is set in the DBTP register.

The following boundary conditions have to be considered for the transmitter delay compensation implement in the MCAN:

- The sum of the measured delay from the m_can_tx to the CAN1_RD and the configured transmitter delay compensation offset bit field in the TDCR register has to be less than 6 bit times in the data phase.
- The sum of the measured delay from the m_can_tx to the CAN1_RD and the configured transmitter delay compensation offset bit field in the TDCR register has to be less or equal 127 MCAN clock periods. In case this sum exceeds 127 MCAN clock periods, the maximum value of 127 MCAN clock periods is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter that stops checking of receive bits at the SSPs.

41.14.2.2 Transmitter delay compensation measurement

If the transmitter delay compensation is enabled by setting the TDC bit in the DBTP register, the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the MCAN_RX of the transmitter. The resolution of this measurement is one MCAN clock periods.

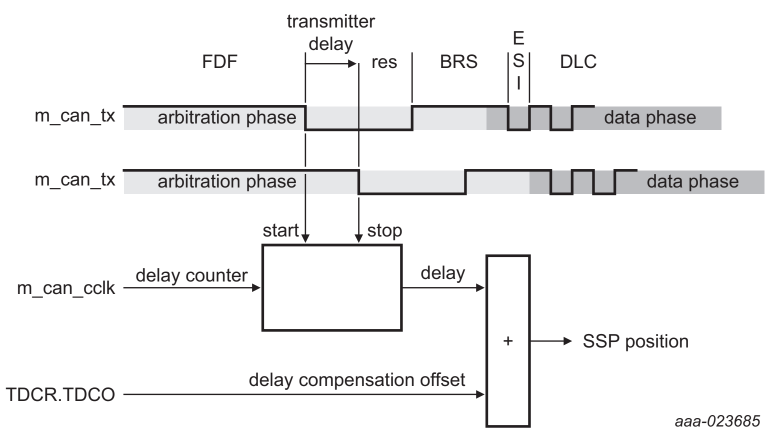


Fig 139. Transmitter delay measurement

To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in a too early SSP position, the use of a transmitter delay compensation filter window can be enabled by programming the TDCF bit field in the TDCR register. This defines a minimum value for the SSP position. Dominant edges on CAN1_RD that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least equal to the TDCF bit field and m_can_rx is low.

41.14.3 Disabled automatic retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the MCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled using the DAR bit field in the CCCR register.

41.14.3.1 Frame transmission in DAR mode

In DAR mode, all transmissions are automatically cancelled after they started on the CAN bus. A Tx buffer’s Tx request pending bit in the TXBRP register is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx buffer transmission occurred bit in the TXBTO register is set.
  - Corresponding Tx buffer cancellation finished bit in the TXBCF register is not set.

- Successful transmission in spite of cancellation:
  - Corresponding Tx buffer transmission occurred bit in the TXBTO register is set.
  - Corresponding Tx buffer cancellation finished bit in the TXBCF register is set.
- Arbitration lost or frame transmission disturbed:
  - Corresponding Tx buffer transmission occurred bit in the TXBTO register is not set.
  - Corresponding Tx buffer cancellation finished bit in the TXBCF register is set.

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx event FIFO element is written with event type = ‘10’ (transmission in spite of cancellation).

41.14.4 Timestamp generation

For timestamp generation, the MCAN supplies a 16-bit wrap-around counter. The TCP bit field in the TSCC register contains a prescaler that can be configured to clock the counter in multiples of CAN bit times (1…16). The counter is readable via the TSC bit field in the TSCV register. A write access to the TSCV register resets the counter to zero.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx buffer / Rx FIFO (RXTS register) or Tx Event FIFO (TXTS register) element.

By programming the TSS bit in the TSCC register, the 16-bit external timestamp counter can be used.

41.14.5 Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the MCAN supplies a 16-bit timeout counter. It operates as a down-counter and uses the same prescaler controlled by TCP bit field in the TSCC register. The timeout counter is configured via the TOCC register. The actual counter value can be read from TOC bit field in the TSCV. The timeout counter can only be started while the INIT bit in the CCCR register is cleared. It is stopped when the INIT bit is set. For example, when the M_CAN enters a buf_off state.

The operation mode is selected by the TOS bit field in the TOCC register. When operating in continuous mode, the counter starts when the INIT bit is cleared. A write to the TOCV register presets the counter to the value configured by TOP bit field in the TOCC register and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP bit field in the TOCC register. Down-counting is started when the first FIFO element is stored. Writing to TOCV register has no effect.

When the counter reaches zero, the TOO interrupt flag in the IR register is set. In continuous mode, the counter is immediately restarted at the value in the TOP bit field in the TOCC register.

Remark: The clock signal for the timeout counter is derived from the CAN core’s sample point signal. Therefore, the point in time where the timeout counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN core. If the bit rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

41.14.6 Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to the Rx buffers or to one of the two Rx FIFOs, and Rx FIFO's put and get indices.

41.14.6.1 Acceptance filtering

The MCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx buffer or to Rx FIFO 0,1. For acceptance filtering, each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as a range filter, filter for one or two dedicated IDs, or classic bit mask filter.
- Each filter element is configurable for acceptance or rejection filtering.
- Each filter element can be enabled or disabled individually.
- Filters are checked sequentially, execution stops with the first matching filter element.

The related configuration registers are:

- Global filter configuration (GFC).
- Standard ID filter configuration (SIDFC).
- Extended ID filter configuration (XIDFC.
- Extended ID AND mask (XIDAM).

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1.
- Store received frame in Rx buffer.
- Store received frame in Rx buffer and generate pulse at filter event pin.
- Reject received frame.
- Set high priority message interrupt flag (HPM bit in the IR register).
- Set high priority message interrupt flag and store the received frame in FOF 0 or FIFO 1.

Acceptance filtering is started after the complete identifier is received. After acceptance filtering has completed, and if a matching Rx buffer or Rx FIFO has been found, the message handler starts writing the received message data in 32 bit chunks to the matching Rx buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the affected Rx buffer or Rx FIFO:

- Rx buffer
  - New data flag of matching Rx buffer is not set, but the Rx buffer is partially overwritten with received data. For error types, see the LEC and DLEC bit field in the PSR register.

- Rx FIFO
  - Put index of matching Rx FIFO is not updated, but the related Rx FIFO element is partially overwritten with received data. For error types, see the LEC and DLEC bit field in the PSR register. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in Section X have to be considered.

Remark: When an accepted message is written to one of the two Rx FIFOs, or into an Rx buffer, the unmodified received identifier is stored independent of the filters used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

41.14.6.1.1 Range filter

The filter matches for all received frames with message IDs in the range defined by the SFID1/SFID2 resp. EFID1/EFID2.

There are two possibilities when range filtering is used together with extended frames:

EFT = “00”: the message ID of received frames is ANDed with the extended ID AND mask register before the range filter is applied.

EFT = “11”: the extended ID AND mask register is not used for range filtering.

41.14.6.1.2 Filter for specific IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element has to be configured with SFID1= SFID2 resp. EFID1= EFID2.

41.14.6.1.3 Classic bit mask filter

Classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering, SFID1/EFID1 is used as message ID filter, while SFID2/EFID2 is used as filter mask.

A 0 bit at the filter mask will mask out the corresponding bit position of the configured ID filter. For example, the value of the received message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received message ID where the corresponding mask bits are 1 are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are zero, all message IDs match.

41.14.6.1.4 Standard message ID filtering

Figure 140 shows the flow of standard message ID filtering (11-bit identifier).

Controlled by the GFC and SIDFC registers, the message ID, remote transmission request bit (RTR), and the identifier extension bit (IDE) of received frames are compared against the list of configured filter elements.

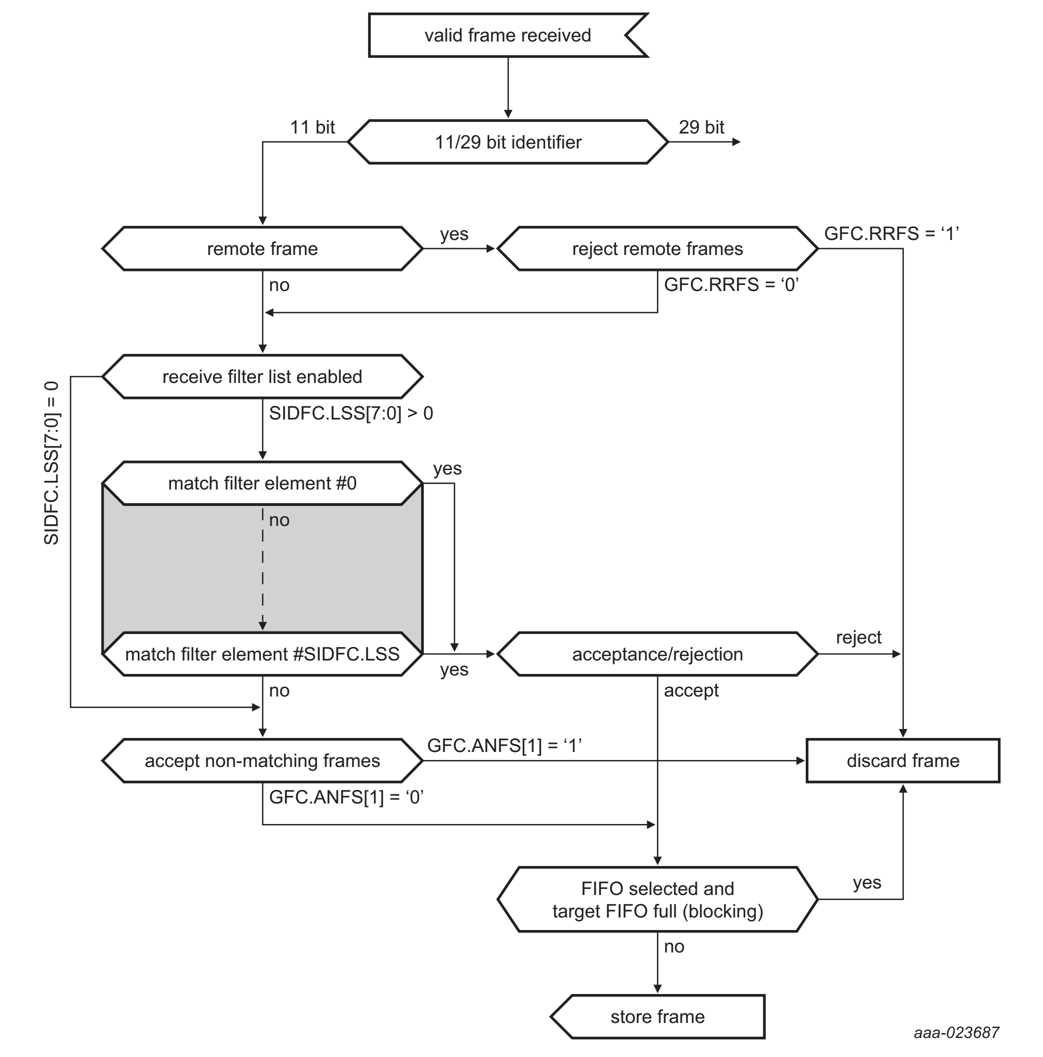


Fig 140. Standard message ID filter path

41.14.6.1.5 Extended message ID filtering

Figure 141 shows the flow of standard message ID filtering (11-bit identifier). The standard message ID filter element is described in Section 41.14.6.1.4 "Standard message ID filtering".

Controlled by the GFC and SIDFC registers, the message ID, remote transmission request bit (RTR), and the identifier extension bit (IDE) of received frames are compared against the list of configured filter elements.

The XIDAM register is ANDed with the received identifier before the filter list is execute.

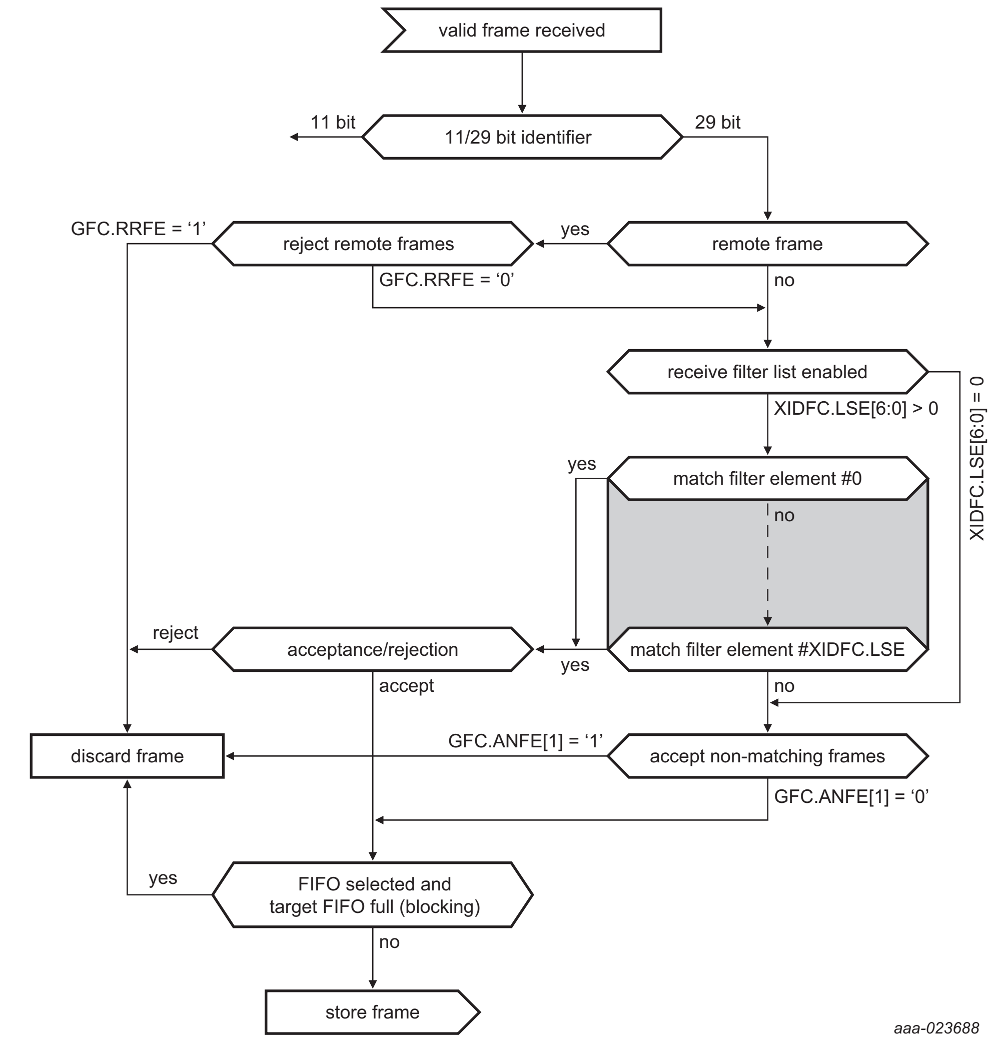


Fig 141. Extended message ID filter path

41.14.6.2 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via the RXF0C and RXF1C registers.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see Section 41.14.6.1 "Acceptance filtering". The Rx FIFO element is described in Section 41.9 "Rx buffer and FIFO element".

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by Rx FIFO watermark bit fields in the RX FIFO configuration registers, the Rx FIFO watermark interrupt flag in the IR register is set. When the Rx FIFO put index reaches the Rx FIFO get index an Rx FIFO full condition is signaled by Rx FIFO full bit fields in the Rx FIFO status registers. In addition, the Rx FIFO full interrupt flag in the IR register is set.

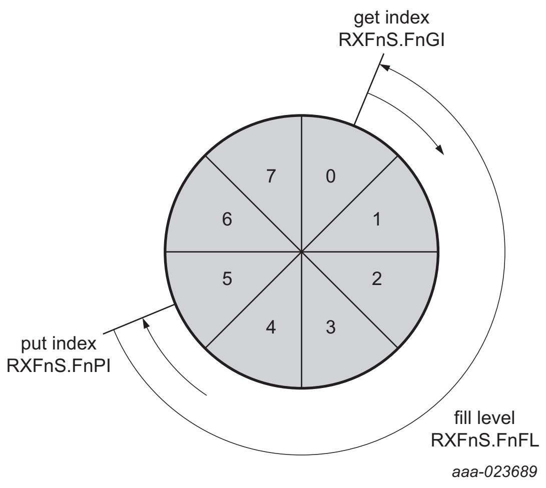


Fig 142. Rx FIFO status

When reading from an Rx FIFO, the Rx FIFO get index bit field in the Rx FIFO status register * FIFO element size has to be added to the corresponding Rx FIFO start address in the Rx FIFO configuration register.

Table 808. Rx buffer / FIFO element size

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field (bytes)	FIFO element size (RAM words)
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

41.14.6.2.1 Rx FIFO blocking mode

The Rx FIFO blocking mode is configured in the RX FIFO configuration registers. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (Rx FIFO put index = the Rx FIFO get index), no more messages are written in the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled when the Rx FIFO full bit fields are set. In addition the Rx FIFO full interrupt flag in the IR register is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signaled in the Rx FIFO status register. In addition, the Rx FIFO lost interrupt flag in the IR register is set.

41.14.6.2.2 Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured in the Rx FIFO configuration register.

When an Rx FIFO full condition (Rx FIFO put index = the Rx FIFO get index) is signaled by the Rx FIFO full bit fields in Rx FIFO status register, the next message accepted by the FIFO will overwrite the oldest FIFO message. The put and get indices are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for this is that a received message is written to the message RAM (put index) while the MCU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the MCU accesses the Rx FIFO. Figure 143 shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

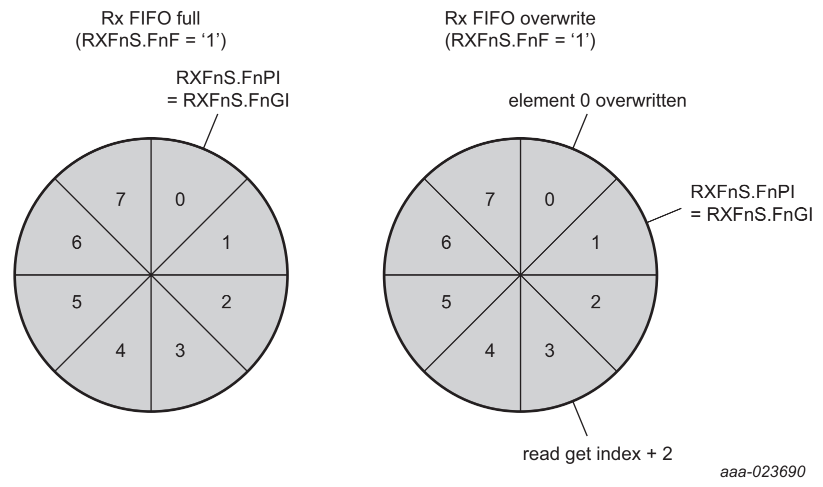


Fig 143. Rx FIFO overflow handling

41.14.6.2.3 Dedicated Rx buffers

The MCAN supports up to 64 dedicated Rx buffers. The start address of the dedicated Rx buffer section is configured via the RBSA bit in the RXBC register.

For each Rx buffer a Standard or extended message ID filter element with SFEC / EFEC = “111” and SFID2 / EFID2[10:9] = “00” has to be configured.

After a received message is accepted by a filter element, the message is stored into the Rx buffer in the message RAM that is referenced by the filter element. The format is the same as an Rx FIFO element. In addition, the DRX interrupt flag is set in the IR register.

Table 809. Example filter configuration for Rx buffers

Filter element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the message RAM, the respective new data flags are set in the NDAT registers. As long as the new data flag is set, the respective Rx buffer is locked against updates from received matching frames. The new data flags have to be reset by writing a 1 to the respective bit position.

While an Rx buffer’s new data flag is set, a message ID filter element referencing this specific Rx buffer will not match, causing the acceptance filtering to continue. Subsequent message ID filter elements may cause the received message to be stored into another Rx buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

41.14.6.2.4 Rx buffer handling

- Reset the DRX interrupt flag in the IR register.
- Read data from the NDAT registers.
- Read messages from the message RAM.
- Reset the NDAT flags of processed messages.

41.14.7 Tx handling

The Tx handler handles transmission requests for the dedicated Tx buffers, the Tx FIFO, and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get Indices, and the Tx event FIFO. Up to 32 Tx buffers can be set up for message transmission. The CAN mode for transmission (classic CAN or CAN FD) can be configured separately for each Tx buffer element. The Tx buffer element is described in Section 41.10 “Tx buffer element”. Figure 19 below describes the possible configurations for frame transmission.

Table 810. Possible configurations for frame transmission

CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
ignored	0	ignored	ignored	Classic CAN
0	1	0	ignored	Classic CAN
0	1	1	ignored	FD without bit rate switching
1	1	0	ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD without bit rate switching

Remark: AUTOSAR requires at least three Tx queue buffers and support for transmit cancellation

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the TXBRP register is updated, or when a transmission has been started.

41.14.7.1 Transmit pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

For example, if CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by the TXP bit in the CCCR register. If the bit is set, the MCAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. By default, transmit pause is disabled in the CCCR register.

This feature breaks up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

41.14.7.2 Dedicated Tx buffers

Dedicated Tx buffers are intended for message transmissions under complete control of the Host CPU. Each dedicated Tx buffer is configured with a specific message ID. In case multiple Tx buffers are configured with the same message ID, the Tx buffer with the lowest buffer number is transmitted first.

If the data section id updated, a transmission is requested by the add request bit fields in the TXBAR register. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx queue and externally with messages on the CAN bus, and are sent out according to their message ID.

A dedicated Tx buffer allocates element size 32-bit words in the message RAM. Therefore the start address of a dedicated Tx buffer in the message RAM is calculated by adding transmit buffer index * element size to the Tx buffer start address stored in the TBSA bit field in the TXBC register.

Table 811. Tx buffer / FIFO / queue element size

TXESC.TBDS[2:0]	Data Field (bytes)	Element size (RAM words)
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8

Table 811. Tx buffer / FIFO / queue element size *…continued*

TXESC.TBDS[2:0]	Data Field (bytes)	Element size (RAM words)
101	32	10
110	48	14
111	64	18

41.14.7.3 Tx FIFO

Tx queue operation is configured by setting the TFQM bit in the TXBC register. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority). In case multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages must be written to the Tx buffer referenced by the Tx FIFO queue put index bit field in the TXFQS register. An add request cyclically increments the put index to the next free Tx buffer. In case the Tx queue is full, the put index is not valid and no more message should be written to the Tx queue until at least one of the requested messages is sent out or a pending transmission request is cancelled.

The application may use the TXBRP register instead of the put index and may place messages to any Tx buffer without pending transmission request.

A dedicated Tx buffer allocates element size 32-bit words in the message RAM. Therefore, the start address of a dedicated Tx buffer in the message RAM is calculated by adding transmit buffer index * element size to the Tx buffer start address stored in the TBSA bit field in the TXBC register.

41.14.7.4 Tx queue

Tx queue operation is configured by programming the TFQM bit field to a value of one. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority). In case that multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages must be written to the Tx buffer referenced by the put index stored in the TFQPI bit field in the TXFQS register. An add request cyclically increments the put index to the next free Tx buffer. In case the Tx queue is full (if the TFQF bit field is set in the TXFQS register), the put index is not valid and no more messages should be written to the Tx queue until at least one of the requested messages is sent out or a pending transmission request has been cancelled.

The application may use the TXBRP register instead of the put index and may place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocated element size 32-bit words in the message RAM. Therefore, the start address of the next available (free) Tx queue buffer is calculated by adding Tx FIFO/queue put index sorted in the TFQPI bit field in the TXFQS register * element size to the Tx buffer start address stored in the TBSA bit field in the TXBC register.

41.14.7.5 Mixed dedicated Tx buffers / Tx FIFO

The Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx FIFO. The number of dedicated Tx buffers is configured by the NDTB bit field in the TXBC register. The number of Tx buffers assigned to the Tx FIFO is configured by the TFQS bit field in the TXBC register. In case the TFQS bit field is programmed to zero, only dedicated buffers are used.

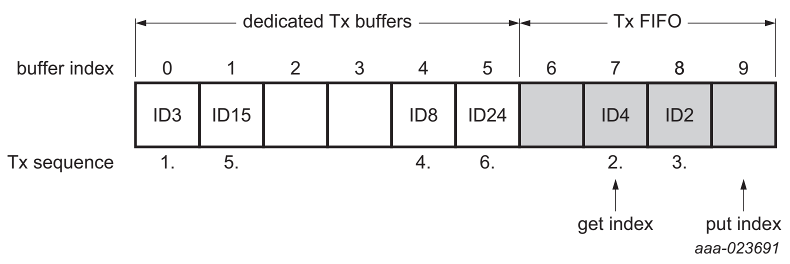


Fig 144. Example of mixed configuration dedicated Tx buffers / Tx FIFO

Tx prioritization:

- Scan dedicated Tx buffers and oldest pending Tx FIFO buffer (referenced by the TFGI bit field in the TXFS register).
- Buffer with lowest message ID gets highest priority and is transmitted next.

41.14.7.6 Mixed dedicated Tx buffers / Tx queue

The Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx queue. The number of dedicated Tx buffers is configured by the NDTB bit field in the TXBC register. The number of Tx queue buffers is configured by the TFQS bit field in the TXBC register. In case the TFQS bit field is programmed with a value of zero, only dedicated Tx buffers are used.

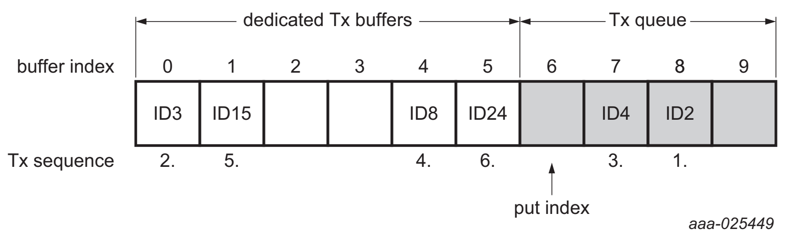


Fig 145. Example of mixed configuration dedicated Tx buffers / Tx queue

Tx prioritization:

- Scan all Tx buffers with activated transmission requests.
- Tx buffer with lowest message ID gets highest priority and is transmitted next.

41.14.7.7 Transmit cancellation

The MCAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx buffer or a Tx queue buffer, set the corresponding bit in the TXBCR register. Transmit cancellation is not intended for Tx FIFO operation. Successful cancellation is signaled by setting the corresponding bit in the TXBCF register.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding bit in the TXBRP register remains set as long as the transmission is in progress. If the transmission was successful, the corresponding bits in the XBTO and TXBCF register are set. If the transmission was not successful, it is not repeated and only the corresponding bit in the TXBCF register is set.

Remark: In case a pending transmission is cancelled immediately before this transmission could have started, there follows a short time window where no transmission has started even if another message is also pending in this node. This may enable another node to transmit a message, which may have a lower priority than the second message in this node.

41.14.7.8 Tx event handling

To support Tx event handling the MCAN has implemented a Tx event FIFO. After the MCAN has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO can be configured to a maximum of 32 elements. The Tx event FIFO element is described in Section 41.11.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling, that is, a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by the TEFF interrupt flag in the IR register, no more elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and the TEFL interrupt flag is set in the IR register.

To avoid a Tx event FIFO overflow, the Tx event FIFO watermark can be used. When the Tx event FIFO fill level reaches the Tx event FIFO watermark configured by the EFWM bit field in the TXEFC register, the TEFW interrupt flag is set in the IR register.

When reading from the Tx event FIFO, two times the Tx event FIFO get index value stored in the EFGI bit field in the TXEFS register has to be added to the Tx event FIFO start address stored in the EFSA bit field in the TXEFC register.

41.14.8 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge. Writing to the FIFO acknowledge index will set the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element is read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO Acknowledge Index.
- When a sequence of elements are read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value: Index of the last element read), to update the get index of the FIFO.

Take special care when reading FIFO elements in an arbitrary order (that is, not using get index). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case the acknowledge index of the FIFO should not be written because this would set the get index to a wrong position and also alter the fill level of the FIFO. In this case some of the older FIFO elements would be lost.

Remark: The application has to ensure that a valid value is written to the FIFO acknowledge index. The MCAN does not check for erroneous values.

UM11424

Chapter 42: LPC55S0x/LPC550x CRC engine

Rev. 1.5 — 21 December 2023 User manual

42.1 How to read this chapter

The CRC engine is available on LPC55S0x/LPC550x devices.

42.2 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.
- Accept any size of data width per write: 8, 16 or 32-bit.
  - 8-bit write: 1-cycle operation.
  - 16-bit write: 2-cycle operation (8-bit x 2-cycle).
  - 32-bit write: 4-cycle operation (8-bit x 4-cycle).

42.3 Basic configuration

Set the CRC bit in the AHBCLKCTRL0 register. See Section 4.5.16 "AHB clock control 0" to enable the clock to the CRC engine.

42.4 Pin description

The CRC engine has no configurable pins.

42.5 General description

The Cyclic Redundancy Check (CRC) generator with programmable polynomial settings supports several commonly used CRC standards.

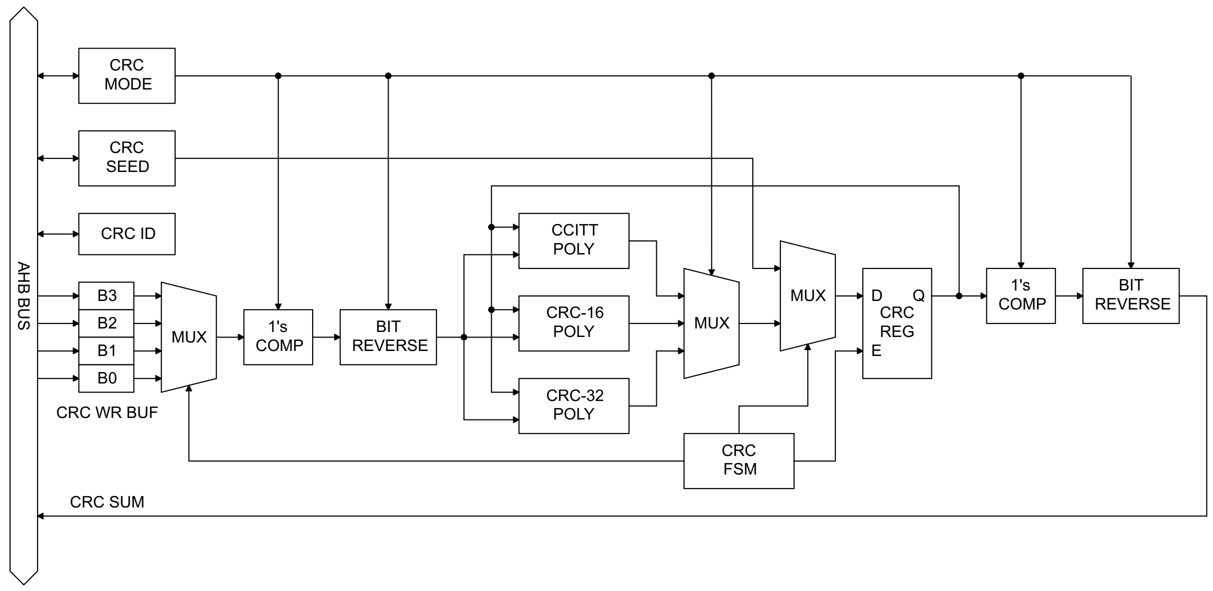


Fig 146. CRC block diagram

42.6 Register description

Table 812. Register overview: CRC engine (base address = 0x4009 5000)

Name	Access	Offset	Description	Reset value	Section
MODE	R/W	0x000	CRC mode register.	0x0000 0000	42.6.1
SEED	R/W	0x004	CRC seed register.	0x0000 FFFF	42.6.2
SUM	RO	0x008	CRC checksum register.	0x0000 FFFF	42.6.3
WR_DATA	WO	0x008	CRC data register.	-	42.6.4

42.6.1 CRC mode register

Table 813. 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 813. 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

42.6.2 CRC seed register

Table 814. CRC seed register (SEED, offset = 0x004)

Bit	Symbol	Description	Reset value
31:0	CRC_SEED	A write access to this register will load CRC seed value to CRC_SUM register with selected bit order and 1’s complement pre-processes. **Remark:** A write access to this register will overrule the CRC calculation in progresses.	0x0000 FFFF

42.6.3 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 815. 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

42.6.4 CRC data register

This register is a write-only register containing the data block for which the CRC sum will be calculated.

Table 816. CRC data register (WR_DATA, offset = 0x008)

Bit	Symbol	Description	Reset value
31:0	CRC_WR_DATA	Data written to this register will be taken to perform CRC calculation with selected bit order and 1’s complement pre-process. Any write size 8, 16 or 32-bit are allowed and accept back-to-back transactions.	-

42.7 Functional description

The following sections describe the register settings for each supported CRC standard:

42.7.1 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

42.7.2 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

42.7.3 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

UM11424

Chapter 43: LPC55S0x/LPC550x Trusted Execution Environment

Rev. 1.5 — 21 December 2023 **User manual**

43.1 How to read this chapter

TrustZone for Armv8-M and a trusted execution environment are available on all LPC55S0x/LPC550x devices. This section discusses the basics of TrustZone for Armv8-M and then describes some additional features of the LPC55S0x/LPC550x that extend the TrustZone security foundation to enable a complete trusted execution environment.

43.2 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 43.3.5 “TrustZone configuration example” for details.

43.3 Functional description

43.3.1 TrustZone for Armv8-M

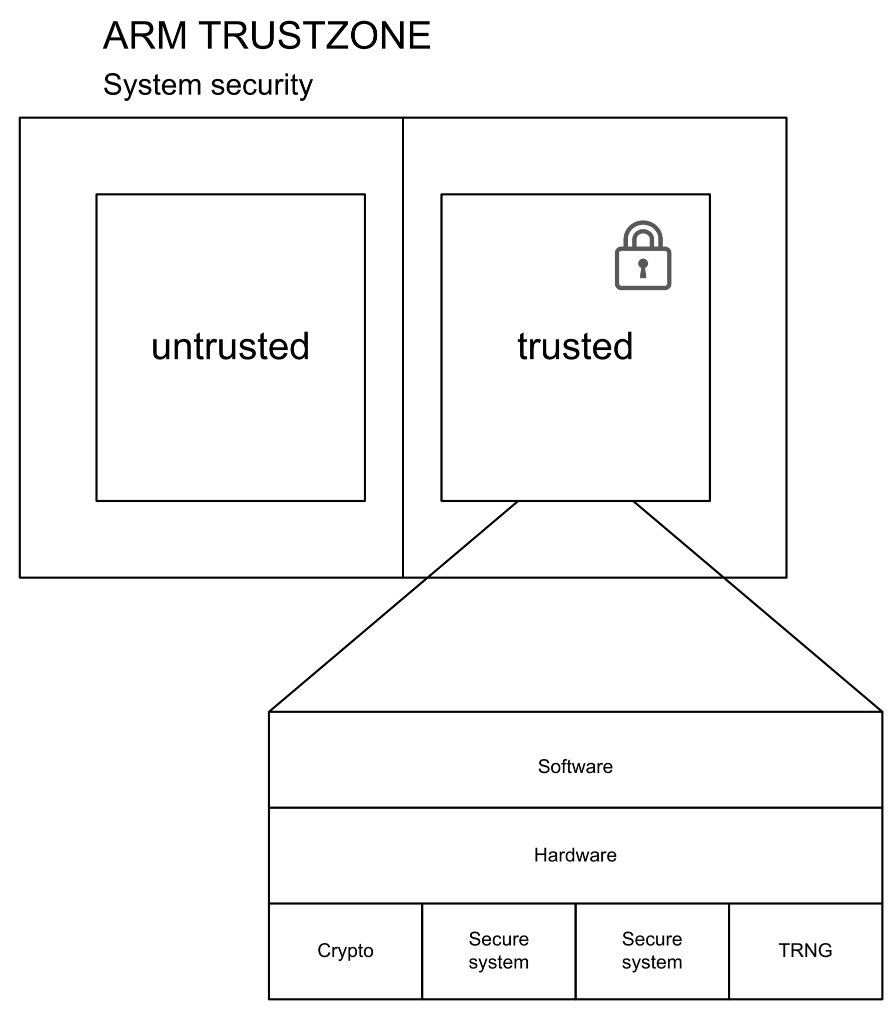


Fig 147. Arm TrustZone system security abstract view

TrustZone for Armv8-M is a new feature available on the 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 the same CPU. This control of security 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.

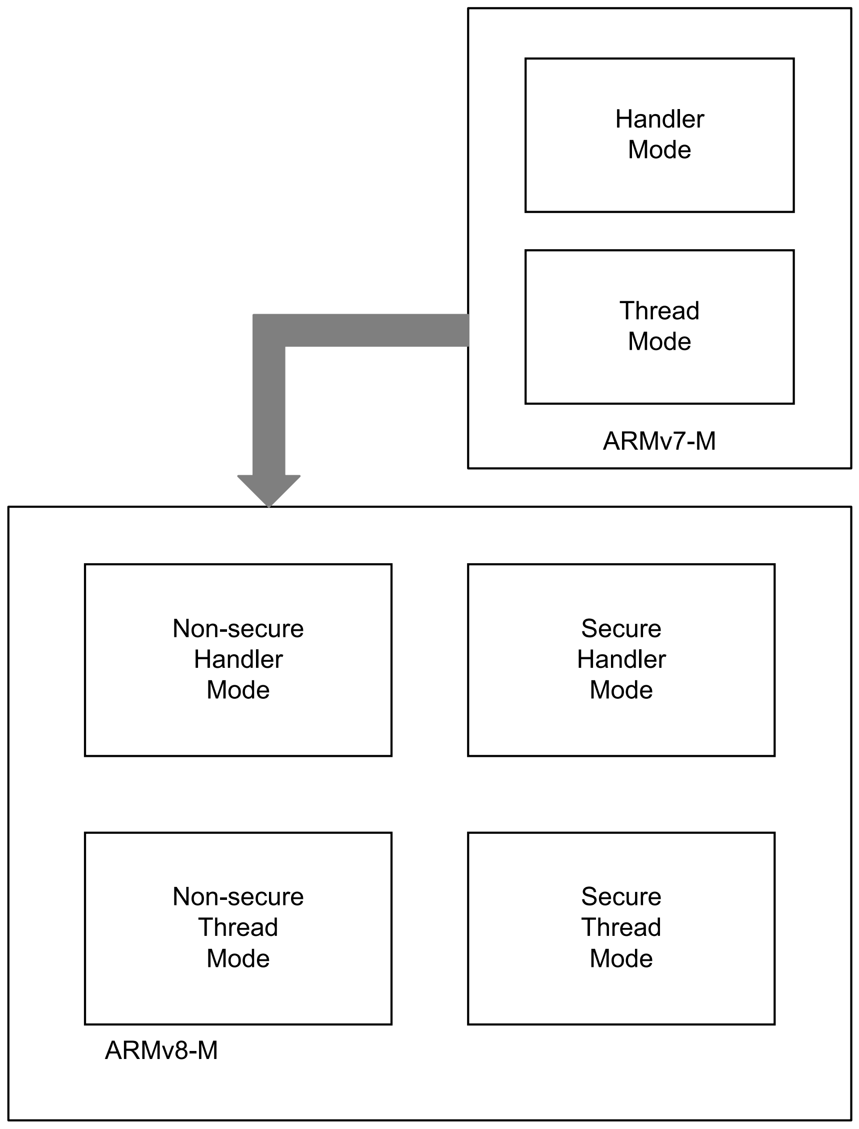


Fig 148. Cortex-M4 (or M3) vs Cortex-M33

More details on TrustZone for ARMv8-M can be found in the “TrustZone® technology for ARM®v8-M architecture version 1.0” document provided through 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.

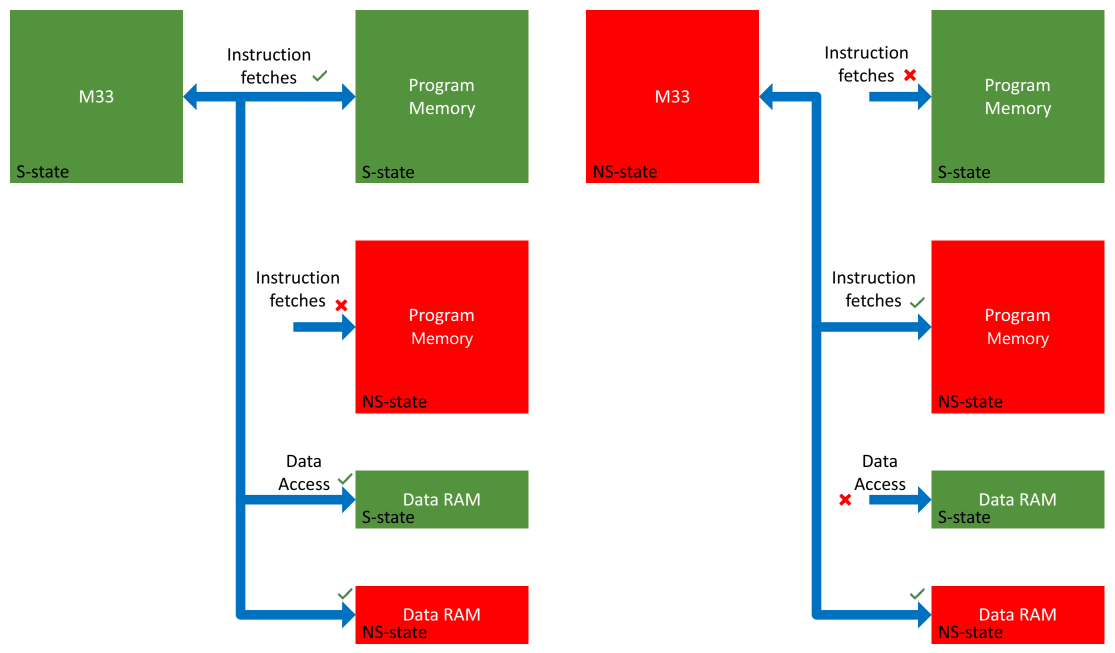


Fig 149. Secure state and Non-secure state view for TrustZone for ARMv8

In summary:

- NS application code *trust* 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 *trust* 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.

43.3.1.1 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). The NSC region lies in S-memory and hence the 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 functions 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 the form of a Secure Gate (SG) instruction. It is placed in the NSC region at the start of a secure function callable from a Non-secure code. When calling into NSC region, the CPU-NS must target an address with the SG instruction. The SG instruction is the only instruction that can be executed when branching from a CPU-NS. On executing the SG instruction, the CPU will change from CPU-NS to the NSC region. 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 ability to access functions inside S-memory.

43.3.2 Attribution units

TrustZone for the 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 150.

A combination of SAU and IDAU assign a specific security attribute (S, NS, or NSC) to a specific address from 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. The secure AHB Controller is discussed in Section 43.3.3.4 “Secure AHB controller”

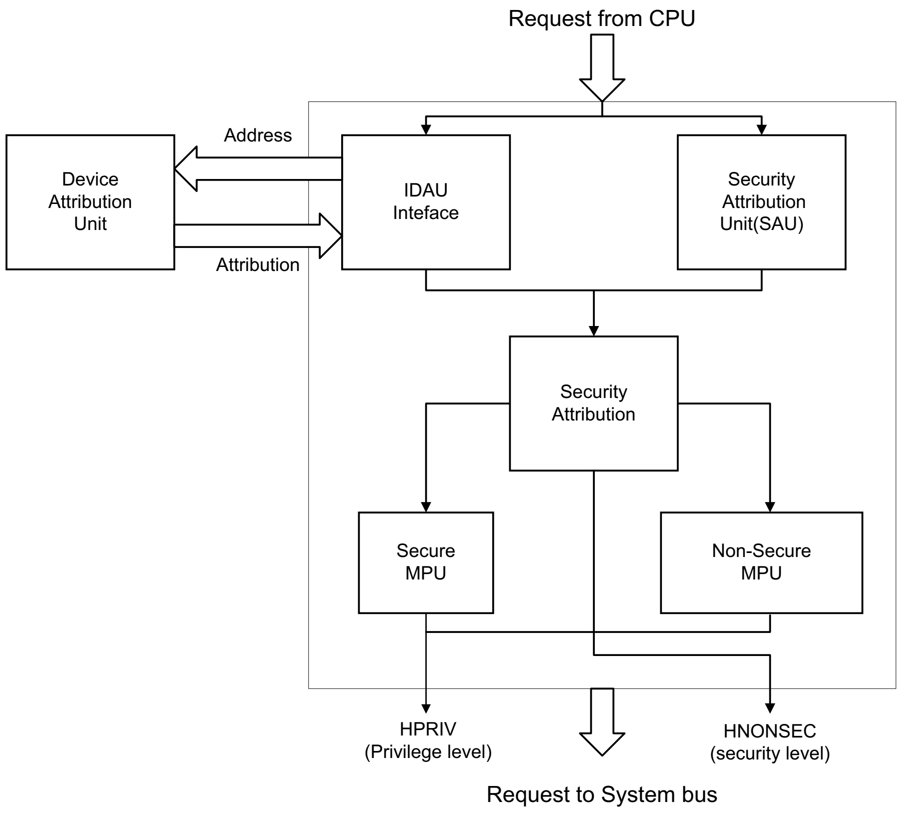


Fig 150. Connection of DAU to CPU0

43.3.2.1 Device Attribution Unit

LPC55S0x/LPC550x implements a simple Attribution unit that divides 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

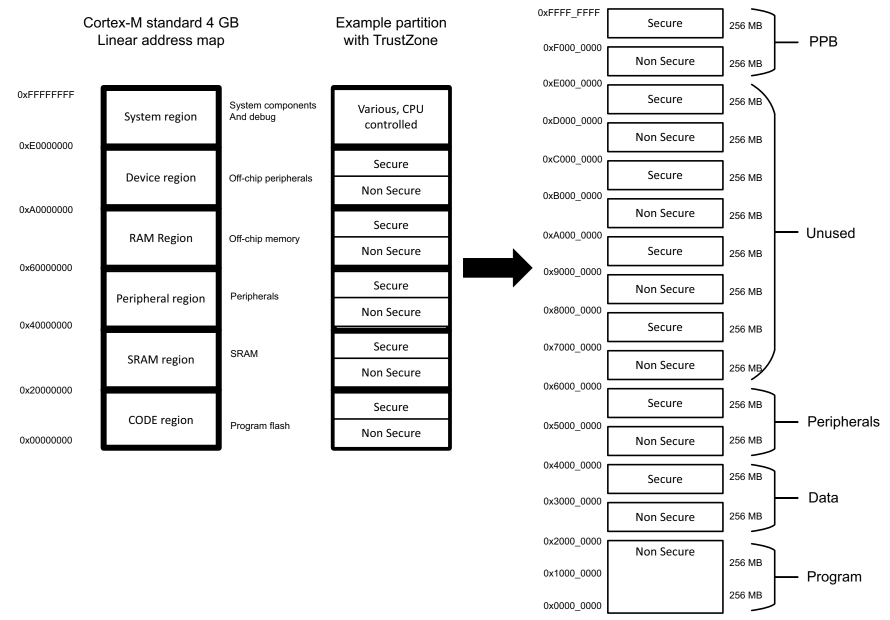


Fig 151. IDAU Implementation

43.3.2.2 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.

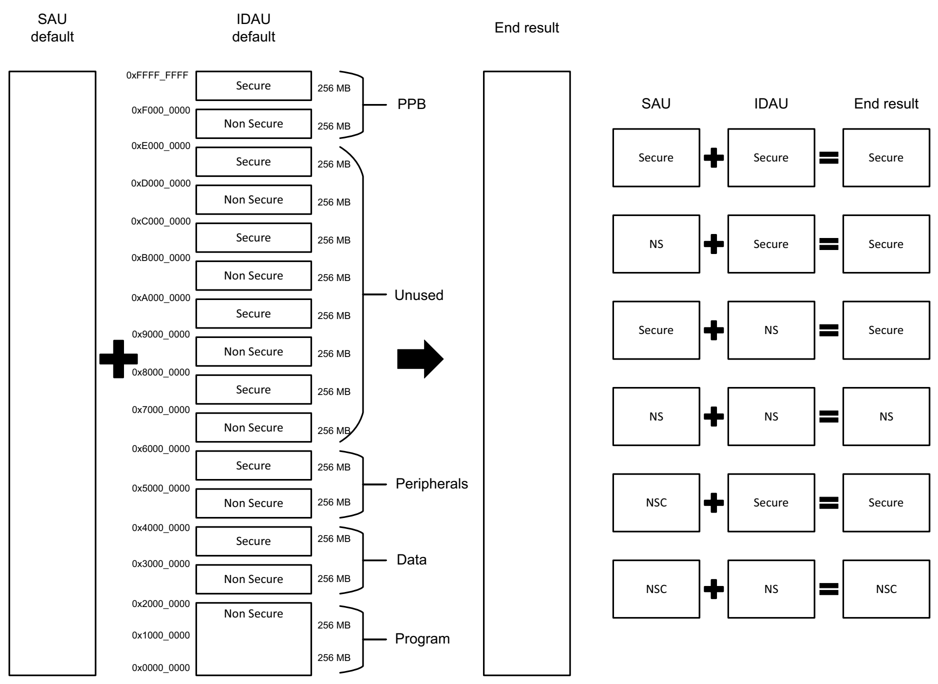


Fig 152. Security attribute definition as combination of SAU and IDAU

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.

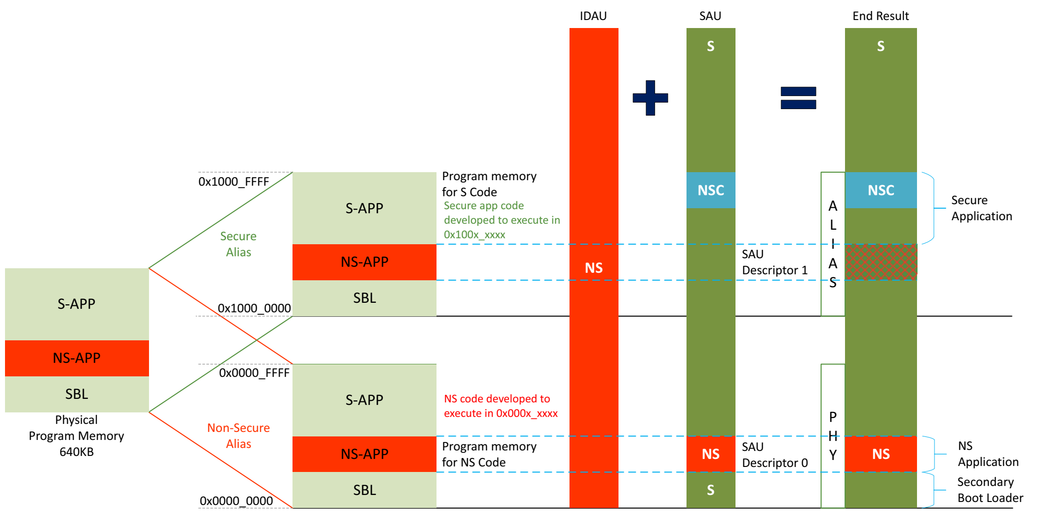


Fig 153. Program memory space aliasing and security attribution example

43.3.2.3 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).

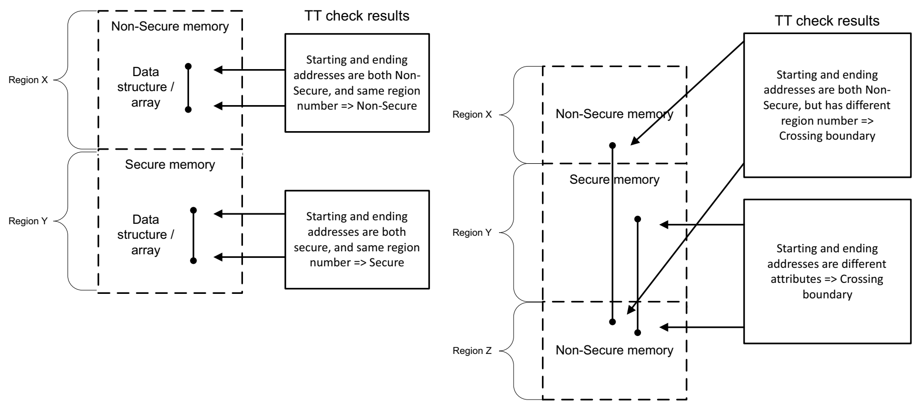


Fig 154. Test Target instruction usage

43.3.3 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 LPC55S0x/LPC550x implements second layer of protection with secure AHB Bus to support secure trusted execution at system-level.

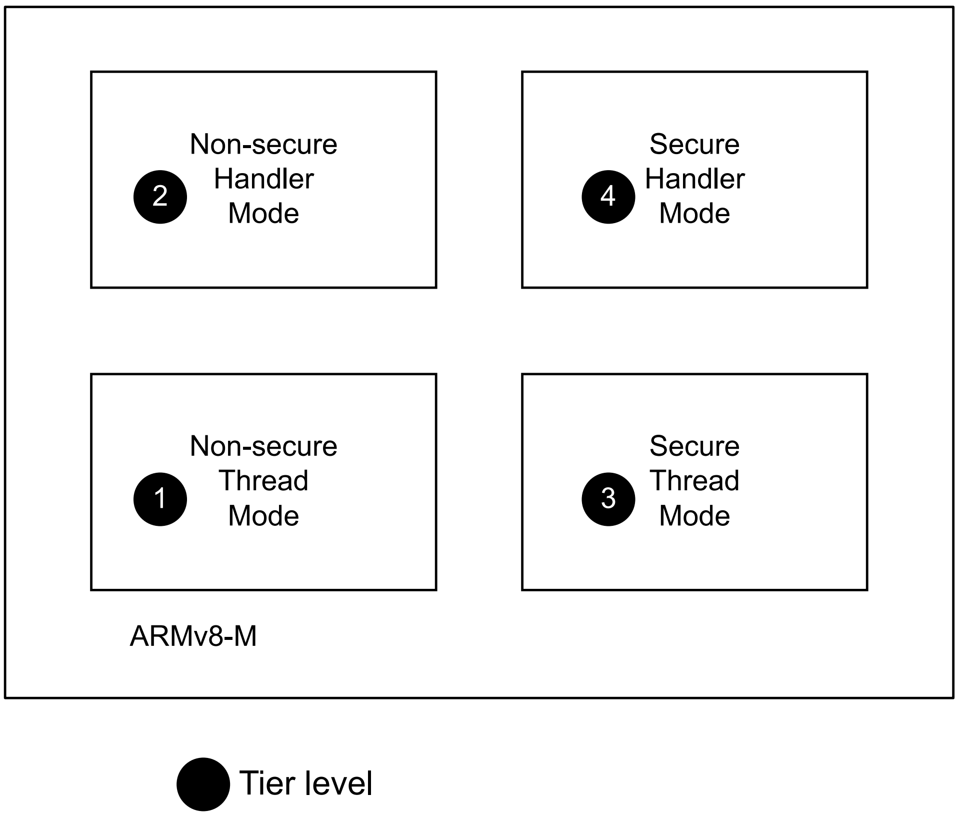


Fig 155. Security tiers as defined using HPRIV and HNONSEC side-band signals

The secure AHB Controller provides access policies for all the bus slaves via checker functions. All masters on LPC55S0x/LPC550x 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.

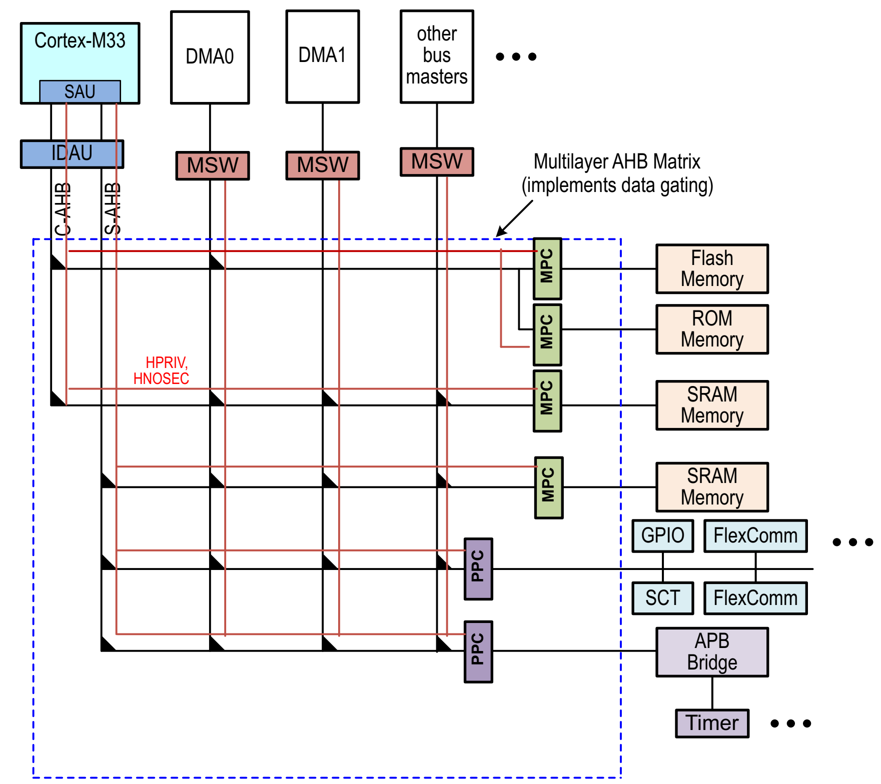


Fig 156. System view with secure AHB bus

These side-band signals create four security tiers as depicted in Figure 156. 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:

43.3.3.1 Memory Protection Checkers (MPC)

On LPC55S0x/LPC550x 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 817. Security tier granularity for on-chip memories

Memory	Total Size	Sub-region size	Sub-regions count
Flash	256 kB	32 kB	8
ROM	128 kB	4 kB	32
SRAM-X	16 kB	4 kB	4
SRAM-0	32 kB	4 kB	8

Table 817. Security tier granularity for on-chip memories *…continued*

Memory	Total Size	Sub-region size	Sub-regions count
SRAM-1	16 kB	4 kB	16
SRAM-2	16 kB	4 kB	16
SRAM-3	16 kB	4 kB	4

43.3.3.2 Peripheral Protection Checkers (PPC)

On LPC55S0x/LPC550x all peripheral on AHB slave port as well as each peripheral on 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 level for security tier assignment and filtering. Each APB bridge has 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.

43.3.3.3 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 LPC55S0x/LPC550x. 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: 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. A protective feature within the module guards against malicious intent by restricting access to a master with the same or higher 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.

43.3.3.4 Secure AHB controller

Secure AHB controller is a module on LPC55S0x/LPC550x 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. This enables BootROM to safeguard certain security features and disable the possibility of enabling those dynamically by unintentionally or with malicious intent.

It also supports register programing for GPIO masking. Details are described in Section 43.4 "Register description".

When a security violation is detected, an interrupt is raised by the secure AHB Controller module. It also logs violation information, (such as the address being accessed on a AHB slave port), when the violation occurred as well as access type and security attributes of the master that generated the unauthorized access.

The only application that can write to a secure AHB Controller to configure security attributes is one that has secure and privileged access rights. Hence, from an application perspective, only the highest tier (tier-4) thread from CPU0 can program security attributes for system slaves.

The 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 a deep-power down.

43.3.4 Interrupt, DMA and GPIO: Secure instance and masking

TrustZone on the LPC55S0x/LPC550x provides secure NVIC (NVIC_NS) and non-secure NVIC (NVIC_NS) capabilities. CPU0 has internal programmability to configure any interrupt as a secure interrupt, thus allowing it to be visible to NVIC_S while masking it from NVIC_NS. Refer to the ARM CM33 documents for more details.

The LPC55S0x/LPC550x has two DMA instances where DMA0 offers 23 channels and DMA1 offers 10 channels. Either DMA can be selected as a 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, the DMA masking feature is implemented. DMA masking can be programmed using the appropriate registers as described in Chapter 18 "LPC55S0x/LPC550x Input Multiplexing (INPUTMUX)"or Chapter 22 "LPC55S0x/LPC550x DMA controller".

All digital IO pin states are readable through the 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 LPC55S0x/LPC550x. Any digital I/O that is sensitive to information leakage can be masked using the SEC_GPIO_MASK0/1 registers with an offset of 0xF80-0xF84 in a secure AHB Controller module.

Additionally LPC55S0x/LPC550x also has additional instance of GPIO-HS and GPIO-PINT module on Port0 (0-31). Unlike normal GPIO, this GPIO functionality is implemented as an IOMUX function and is available only if selected using IOCON programing. It can be used as secure GPIO to generate certain input patterns from an external device for secure signaling. More details can be found in Chapter 16 "LPC55S0x/LPC550x General Purpose I/O (GPIO)" and Chapter 20 "LPC55S0x/LPC550x Secure pin interrupt and pattern match (Secure PINT)" chapters.

43.3.5 TrustZone configuration example

The LPC55s0x/LPC550x 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 157 "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.

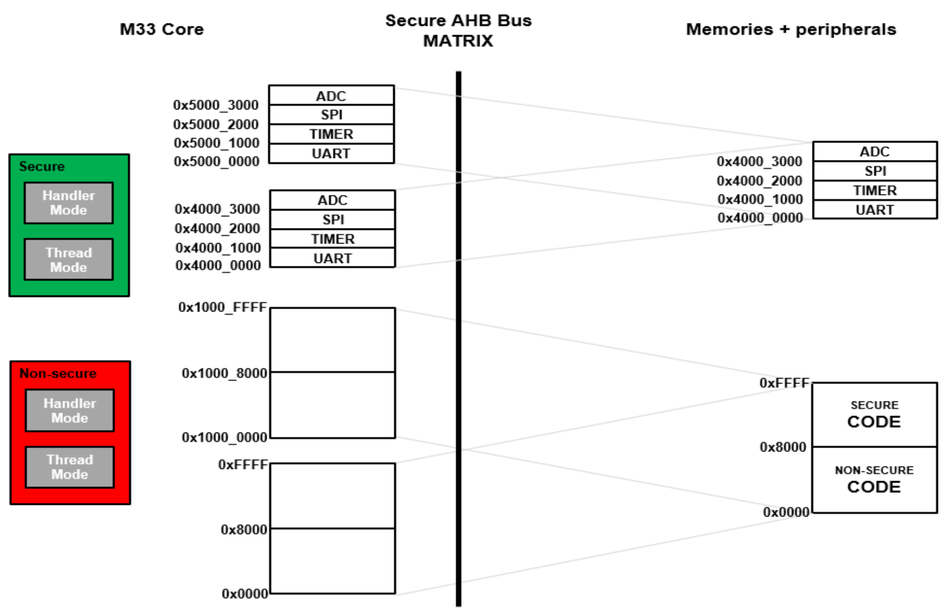


Fig 157. Example of cortex-M33 device with security extension

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 818 "MCU memory layout after TrustZone configuration":

Table 818. 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 819 "Basic SAU configuration".

Table 819. 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 158 "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.

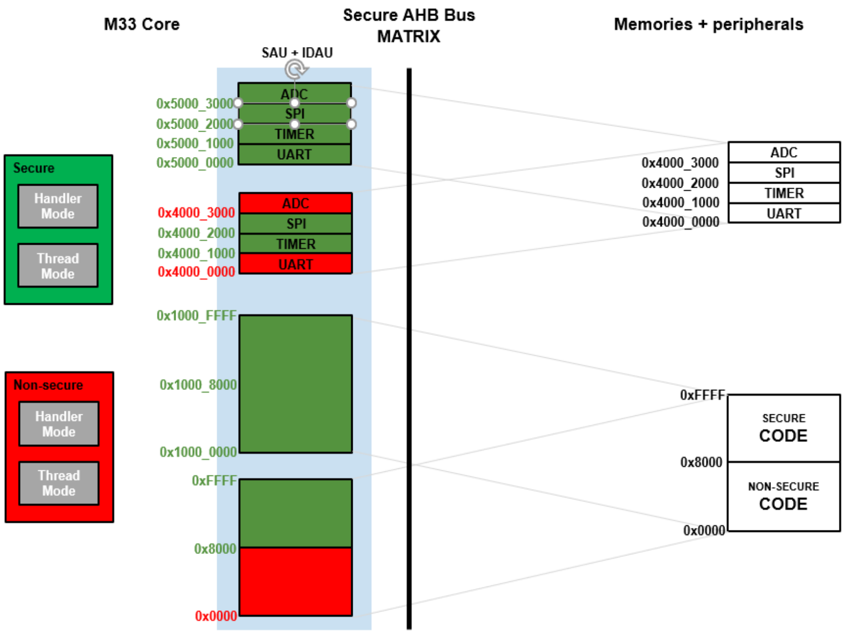


Fig 158. TrustZone isolation after basic SAU configuration

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, 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 820 "Canonical SAU configuration".

Table 820. 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 159 "TrustZone isolation after canonical SAU configuration", configured using canonical SAU configuration..

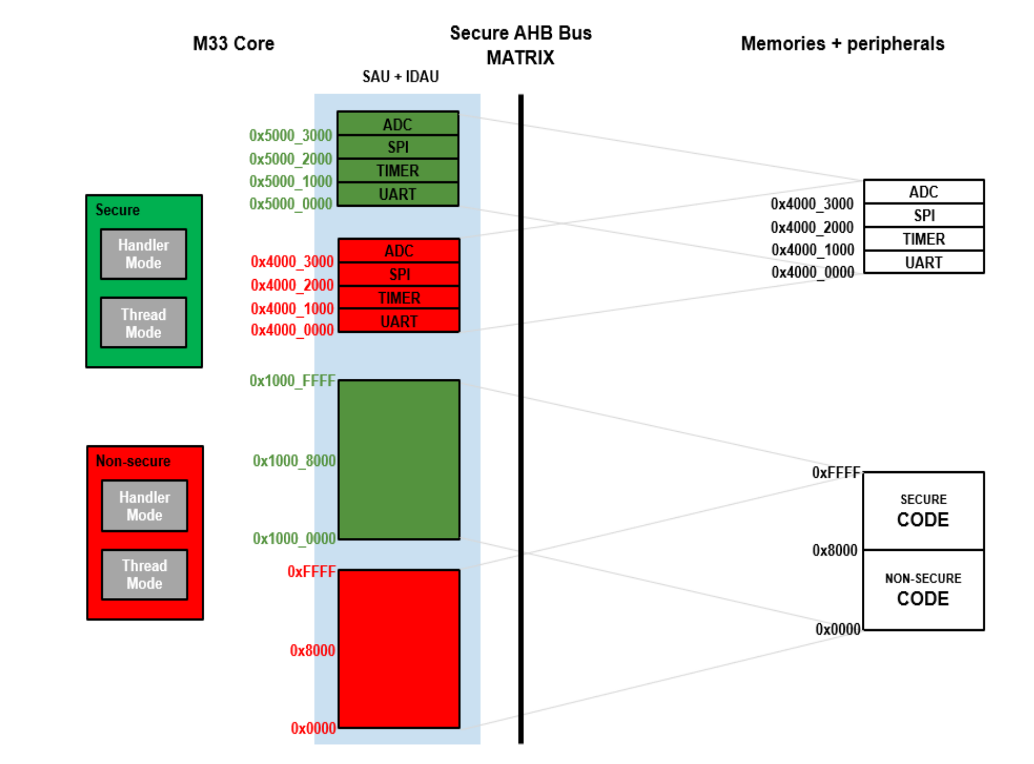


Fig 159. TrustZone isolation after canonical SAU configuration

Comparing Figure 158 "TrustZone isolation after basic SAU configuration" and Figure 159 "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 160 “TrustZone isolation after canonical form of SAU and secure AHB controller configuration”.

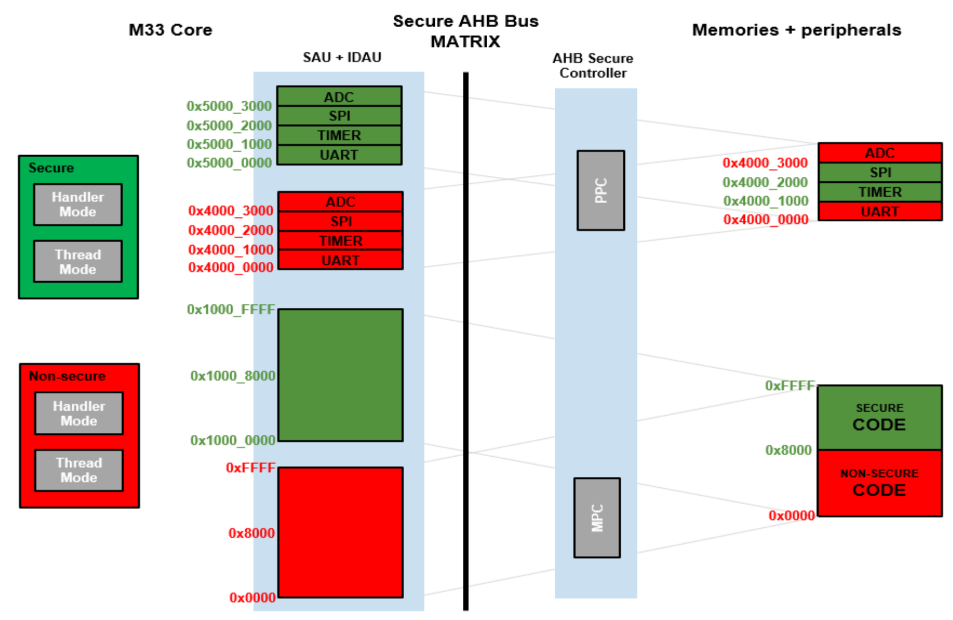


Fig 160. TrustZone isolation after canonical form of SAU and secure AHB controller configuration

After proper AHB secure controller configuration, the TrustZone isolation is fully functional. Compared with the configuration shown in Figure 159 “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 158 “TrustZone isolation after basic SAU configuration”.

The TrustZone example shown in Figure 161 “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 821. 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

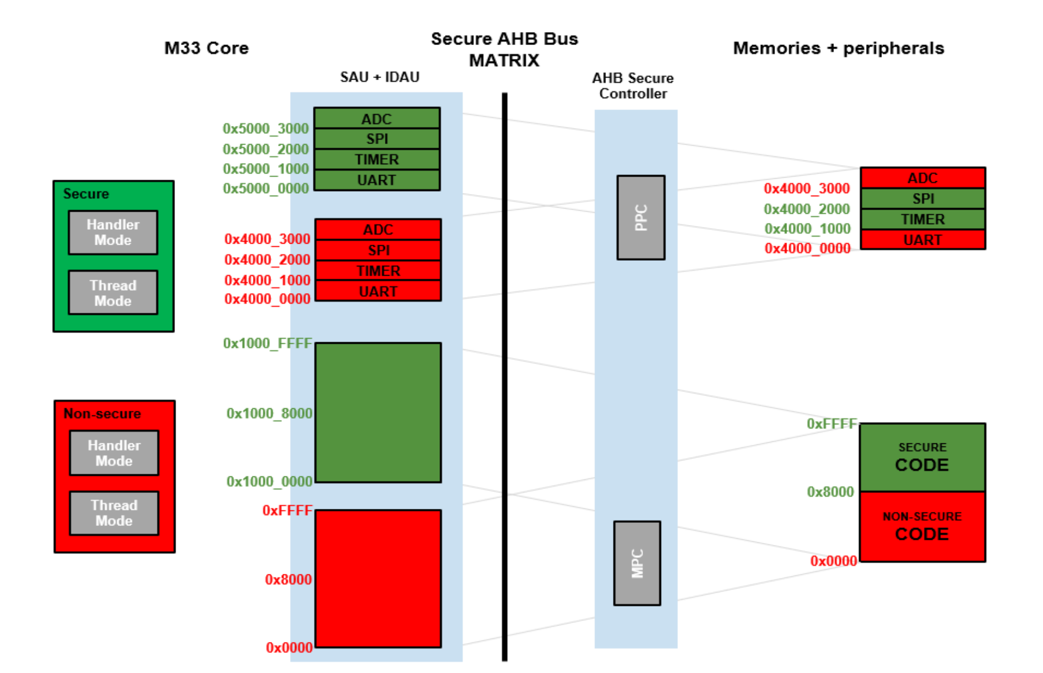


Fig 161. TrustZone isolation after combined SAU and secure AHB controller configuration

43.4 Register description

Table 822. Register overview: AHB_Secure_CTRL (base address = 0x400AC000) [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	43.4.1
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	43.4.2
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.	0x3	43.4.3

Table 822. Register overview: AHB_Secure_CTRL (base address = 0x400AC000) *…continued* [1] [2]

Name	Access	Offset	Description	Reset value	Section
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	43.4.4
SEC_CTRL_ROM_MEM_RULE2	RW	0x28	Security access rules for ROM sector 16 to sector 23, address range 0x0301_0000 - 0x0301_7FFF.	0x0	43.4.5
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	43.4.6
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	43.4.7
SEC_CTRL_RAMX_MEM_RULE0	RW	0x40	Security access rules for sub-regions within RAMX.	0x0	43.4.8
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	43.4.9
SEC_CTRL_RAM0_MEM_RULE0	RW	0x60	Security access rules for sub-regions within RAM0.	0x0	43.4.10
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	43.4.11
SEC_CTRL_RAM1_MEM_RULE0	RW	0x80	Security access rules for sub-regions within RAM1.	0x0	43.4.12
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	43.4.13
SEC_CTRL_RAM2_MEM_RULE0	RW	0xA0	Security access rules for sub-regions within RAM2.	0x0	43.4.14
SEC_CTRL_RAM3_SLAVE_RULE	RW	0xB0	Security access rules for sub-regions within RAM3.	0x0	43.4.15

Table 822. Register overview: AHB_Secure_CTRL (base address = 0x400AC000) *…continued* [1] [2]

Name	Access	Offset	Description	Reset value	Section
SEC_CTRL_RAM3_MEM_RULE0	RW	0xC0	Security access rules for sub-regions within RAM3.	0x0	43.4.16
SEC_CTRL_APB_BRIDGE_SLAVE_RULE	RW	0xD0	Security access rules for APB Bridge slave. This rule supersede more granular security rules as in SEC_CTRL_APB_BRIDGEx_MEM_CTRLx.	0x0	43.4.17
SEC_CTRL_APB_BRIDGE0_MEM_CTRL0	RW	0xE0	Security access rules for individual peripherals on APB Bridge 0. Each peripheral can be assigned independent security level.	0x0	43.4.18
SEC_CTRL_APB_BRIDGE0_MEM_CTRL1	RW	0xE4	Security access rules for individual peripherals on APB Bridge 0. Each peripheral can be assigned independent security level.	0x0	43.4.19
SEC_CTRL_APB_BRIDGE0_MEM_CTRL2	RW	0xE8	Security access rules for individual peripherals on APB Bridge 0. Each peripheral can be assigned independent security level.	0x0	43.4.20
SEC_CTRL_APB_BRIDGE1_MEM_CTRL0	RW	0xF0	Security access rules for individual peripherals on APB Bridge 1. Each peripheral can be assigned independent security level.	0x0	43.4.21
SEC_CTRL_APB_BRIDGE1_MEM_CTRL1	RW	0xF4	Security access rules for individual peripherals on APB Bridge 1. Each peripheral can be assigned independent security level.	0x0	43.4.22
SEC_CTRL_APB_BRIDGE1_MEM_CTRL2	RW	0xF8	Security access rules for individual peripherals on APB Bridge 1. Each peripheral can be assigned independent security level.	0x0	43.4.23
SEC_CTRL_APB_BRIDGE1_MEM_CTRL3	RW	0xFC	Security access rules for individual peripherals on APB Bridge 1. Each peripheral can be assigned independent security level.	0x0	43.4.24
SEC_CTRL_AHB_PORT7_SLAVE0_RULE	RW	0x100	Security access rules for AHB peripherals. Each peripheral can be assigned independent security level.	0x0	43.4.25
SEC_CTRL_AHB_PORT7_SLAVE1_RULE	RW	0x104	Security access rules for AHB peripherals. Each peripheral can be assigned independent security level.	0x0	43.4.26
SEC_CTRL_AHB_PORT8_SLAVE0_RULE	RW	0x110	Security access rules for AHB peripherals. Each peripheral can be assigned independent security level.	0x0	43.4.27
SEC_CTRL_AHB_PORT8_SLAVE1_RULE	RW	0x114	Security access rules for AHB peripherals. Each peripheral can be assigned independent security level.	0x0	43.4.28

Table 822. Register overview: AHB_Secure_CTRL (base address = 0x400AC000) *…continued* [1] [2]

Name	Access	Offset	Description	Reset value	Section
SEC_CTRL_AHB_PORT9_SLAVE0_RULE	RW	0x120	Security access rules for AHB peripherals. Each peripheral can be assigned independent security level.	0x0	43.4.29
SEC_CTRL_AHB_PORT9_SLAVE1_RULE	RW	0x124	Security access rules for AHB peripherals. Each peripheral can be assigned independent security level.	0x0	43.4.30
SEC_CTRL_AHB_SEC_CTRL_MEM_RULE	RW	0x130	Security access rules for AHB peripherals. Each peripheral can be assigned independent security level.	0x0	43.4.31
SEC_VIO_ADDR0	R	0xE00	Most recent security violation address for AHB port ADDR0.	0x0	43.4.32
SEC_VIO_ADDR1	R	0xE04	Most recent security violation address for AHB port ADDR1.	0x0	43.4.33
SEC_VIO_ADDR2	R	0xE08	Most recent security violation address for AHB port ADDR2.	0x0	43.4.34
SEC_VIO_ADDR3	R	0xE0C	Most recent security violation address for AHB port ADDR3.	0x0	43.4.35
SEC_VIO_ADDR4	R	0xE10	Most recent security violation address for AHB port ADDR4.	0x0	43.4.36
SEC_VIO_ADDR5	R	0xE14	Most recent security violation address for AHB port ADDR5.	0x0	43.4.37
SEC_VIO_ADDR6	R	0xE18	Most recent security violation address for AHB port ADDR6.	0x0	43.4.38
SEC_VIO_ADDR7	R	0xE1C	Most recent security violation address for AHB port ADDR7.	0x0	43.4.39
SEC_VIO_ADDR8	R	0xE20	Most recent security violation address for AHB port ADDR8.	0x0	43.4.40
SEC_VIO_ADDR9	R	0xE24	Most recent security violation address for AHB port ADDR9.	0x0	43.4.41
SEC_VIO_MISC_INFO0	R	0xE80	Most recent security violation miscellaneous information for AHB port INFO0.	0x0	43.4.42
SEC_VIO_MISC_INFO1	R	0xE84	Most recent security violation miscellaneous information for AHB port INFO1.	0x0	43.4.43
SEC_VIO_MISC_INFO2	R	0xE88	Most recent security violation miscellaneous information for AHB port INFO2.	0x0	43.4.44
SEC_VIO_MISC_INFO3	R	0xE8C	Most recent security violation miscellaneous information for AHB port INFO3.	0x0	43.4.45
SEC_VIO_MISC_INFO4	R	0xE90	Most recent security violation miscellaneous information for AHB port INFO4.	0x0	43.4.46
SEC_VIO_MISC_INFO5	R	0xE94	Most recent security violation miscellaneous information for AHB port INFO5.	0x0	43.4.47

Table 822. Register overview: AHB_Secure_CTRL (base address = 0x400AC000) *…continued* [1] [2]

Name	Access	Offset	Description	Reset value	Section
SEC_VIO_MISC_INFO6	R	0xE98	Most recent security violation miscellaneous information for AHB port INFO6.	0x0	43.4.48
SEC_VIO_MISC_INFO7	R	0xE9C	Most recent security violation miscellaneous information for AHB port INFO7.	0x0	43.4.49
SEC_VIO_MISC_INFO8	R	0xEA0	most recent security violation miscellaneous information for AHB port INFO8.	0x0	43.4.50
SEC_VIO_MISC_INFO9	R	0xEA4	Most recent security violation miscellaneous information for AHB port INFO9.	0x0	43.4.51
SEC_VIO_INFO_VALID	RW	0xF00	security violation address/information registers valid flags.	0x0	43.4.52
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	43.4.53
SEC_GPIO_MASK1	RW	0xF84	Secure GPIO mask for port 1 pins.	0xFFFFFFFF	43.4.54
SEC_MASK_LOCK	RW	0xFBC	Security General Purpose register access control.	0x2A	43.4.55
MASTER_SEC_LEVEL	RW	0xFD0	Master Secure level register.	0x80000000	43.4.56
MASTER_SEC_ANTI_POL_REG	RW	0xFD4	Master Secure level anti-pole register.	0xBFFFFFFF	43.4.57
CM33_LOCK_REG	RW	0xFEC	Miscellaneous control signals for in CPU0.	0x800002AA	43.4.58
MISC_CTRL_DP_REG	RW	0xFF8	Secure control duplicate register.	0xAAAA	43.4.59
MISC_CTRL_REG	RW	0xFFC	Secure control register.	0xAAAA	43.4.60

[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.

43.4.1 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 823. 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 - 0x0003_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

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

43.4.2 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 824. 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
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 824. Security control flash memory rule0 (SEC_CTRL_FLASH_MEM_RULE0, offset = 0x10) *…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

43.4.3 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 825. 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.	

Table 825. Security access rules for ROM (SEC_CTRL_ROM_MEM_RULE0, offset = 0x20) *…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

43.4.4 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 826. Security access rules for ROM sectors (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

Table 826. Security access rules for ROM sectors (SEC_CTRL_ROM_MEM_RULE1, offset = 0x24) *…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 826. Security access rules for ROM sectors (SEC_CTRL_ROM_MEM_RULE1, offset = 0x24) *…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

43.4.5 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 827. Security access rules for ROM sectors (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.	
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 827. Security access rules for ROM sectors (SEC_CTRL_ROM_MEM_RULE2, offset = 0x28) *…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

43.4.6 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 828. Security access rules for ROM sectors (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.	

Table 828. Security access rules for ROM sectors (SEC_CTRL_ROM_MEM_RULE3, offset = 0x2C) *…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

43.4.7 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 829. 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

43.4.8 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 830. 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

Table 830. Security access rules for RAMX slaves (SEC_CTRL_RAMX_MEM_RULE0, offset = 0x40) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
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

43.4.9 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 831. 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

43.4.10 Security access rules for RAM0 memory

Security access rules for RAM0 sub region 0_0 to 0_7. Each RAM0 sub region is 4 kbytes.

Table 832. 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
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

Table 832. Security access rules for RAM0 slaves (SEC_CTRL_RAM0_MEM_RULE0, offset = 0x60) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
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

43.4.11 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 833. 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 : 0x2000_8000 - 0x2000_BFFF 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: 0x2000_8000 - 0x2000_8FFF

Mem rule 0_1 Address space: 0x2000_9000 - 0x2000_9FFF

Mem rule 0_2 Address space: 0x2000_A000 - 0x2000_AFFF

Mem rule 0_3 Address space: 0x2000_B000 - 0x2000_BFFF

43.4.12 Security access rules for RAM1 memory

Security access rules for RAM1 sub region 0_0 to 0_7. Each RAM1 sub region is 4 kbytes.

Table 834. 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.	
31:14		WO		Reserved.	0x0

43.4.13 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 835. 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 : 0x2000_C000 - 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

RAM2 region is divided into 16 sub-regions, each rule corresponds to a following sub-region address range:

Mem rule 0_0 Address space: 0x2000_C000 - 0x2000_CFFF

Mem rule 0_1 Address space: 0x2000_D000 - 0x2000_DFFF

Mem rule 0_2 Address space: 0x2000_E000 - 0x2000_EFFF

Mem rule 0_3 Address space: 0x2000_F000 - 0x2000_FFFF

43.4.14 Security access rules for RAM2 memory

Security access rules for RAM2 sub region 0_0 to 0_7. Each RAM2 sub region is 4 kbytes.

Table 836. 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.	
31:14		WO		Reserved.	0x0

43.4.15 Security access rules for RAM3 slaves

Security access rules for RAM3 slaves.

Table 837. 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: 0x2001_0000 - 0x2001_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

RAM3 region is divided into 4 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

43.4.16 Security access rules for RAM3

Security access rules for RAM3.

Table 838. Security rules for RAM3 (SEC_CTRL_RAM3_MEM_RULE0, offset 0xC0)

Bit	Symbol	Access	Value	Description	Reset value
1:0	SRAM_ SECT_0 _RULE	RW		Security access rules for the whole RAM3: 0x2001_0000 - 0x2001_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
5:4	SRAM_ SECT_1 _RULE	RW		Security access rules for the whole RAM3: 0x2001_1000 - 0x2001_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

Table 838. Security rules for RAM3 (SEC_CTRL_RAM3_MEM_RULE0, offset 0xC0) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
9:8	SRAM_ SECT_2 _RULE	RW		Security access rules for the whole RAM3: 0x2001_2000 - 0x2001_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		Security access rules for the whole RAM3: 0x2001_3000 - 0x2001_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

RAM3 region is divided into 4 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

43.4.17 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 839. Security control APB bridge slave rule (SEC_CTRL_APB_BRIDGE_SLAVE_RULE, offset = 0xD0)

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

Table 839. Security control APB bridge slave rule (SEC_CTRL_APB_BRIDGE_SLAVE_RULE, offset = 0xD0)

Bit	Symbol	Access	Value	Description	Reset value
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

43.4.18 Secure control APB bridge0 memory control0

Security access rules for APB Bridge 0 peripherals. Each peripheral can have independent security attribute

Table 840. Secure control APB Bridge memory control (SEC_CTRL_APB_BRIDGE0_MEM_CTRL0, offset = 0xE0)

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	-	-		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.	

Table 840. Secure control APB Bridge memory control (SEC_CTRL_APB_BRIDGE0_MEM_CTRL0, offset = 0xE0)

Bit	Symbol	Access	Value	Description	Reset value
19:18	-	-		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
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	-	-		Reserved.	0x0

43.4.19 Secure control APB bridge0 memory control1

Security access rules for APB Bridge 0 peripherals. Each peripheral can have independent security attribute.

Table 841. Secure control APB Bridge0 memory control1 (SEC_CTRL_APB_BRIDGE0_MEM_CTRL1, offset = 0xE4)

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	-	-		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

Table 841. Secure control APB Bridge0 memory control1 (SEC_CTRL_APB_BRIDGE0_MEM_CTRL1, offset = 0xE4)

Bit	Symbol	Access	Value	Description	Reset value
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
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

43.4.20 Secure control APB bridge0 memory control 2

Security access rules for APB Bridge 0 peripherals. Each peripheral can have independent security attribute.

Table 842. Secure control APB Bridge0 memory control2 (SEC_CTRL_APB_BRIDGE0_MEM_CTRL2, offset = 0xE8)

Bit	Symbol	Access	Value	Description	Reset value
11:0	-	-		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	-	-		Reserved.	0x0

43.4.21 Secure control APB bridge1 memory control 0

Security access rules for APB Bridge 1 peripherals. Each peripheral can have independent security attribute.

Table 843. Secure control APB Bridge1 memory control 0(SEC_CTRL_APB_BRIDGE1_MEM_CTRL0, offset = 0xF0)

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 843. Secure control APB Bridge1 memory control 0(SEC_CTRL_APB_BRIDGE1_MEM_CTRL0, offset = 0xF0)

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

43.4.22 Secure control APB bridge1 memory control1

Security access rules for APB Bridge 1 peripherals. Each peripheral can have independent security attribute.

Table 844. Secure Control APB Bridge1 Memory Control1 (SEC_CTRL_APB_BRIDGE1_MEM_CTRL1, offset = 0xF4)

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 844. Secure Control APB Bridge1 Memory Control1 (SEC_CTRL_APB_BRIDGE1_MEM_CTRL1, offset = 0xF4)

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

43.4.23 Secure control APB bridge1 memory control2 register

Security access rules for APB Bridge 1 peripherals. Each peripheral can have independent security attribute. Each APB bridge sector is 4 Kbytes. There are 32 APB Bridge 1 sectors in total.

Table 845. Secure control APB bridge1 memory control2 (SEC_CTRL_APB_BRIDGE1_MEM_CTRL2, offset = 0xF8)

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		Prince.	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	-	-		Reserved.	0x0

43.4.24 Security access rules for APB Bridge 1 peripherals.

Security access rules for APB Bridge 1 peripherals. Each peripheral can have independent security attribute.

Table 846. Secure control APB bridge1 memory control3 register (SEC_CTRL_APB_BRIDGE1_MEM_CTRL3, offset = 0x11C)

Bit	Symbol	Access	Value	Description	Reset value
1:0	-	-		Reserved.	0x0
7:2		WO		Reserved.	0x0
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		PUF.	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	-	-		Reserved.	0x0

43.4.25 Security access rules for AHB peripherals

Security access rules for AHB peripherals on AHB Slave Port P7. Each peripheral can have independent security attribute.

Table 847. Security control AHB0 slave rule (SEC_CTRL_AHB_PORT7_SLAVE0, offset = 0x100)

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
19:16		WO		Reserved.	0x0

Table 847. Security control AHB0 slave rule (SEC_CTRL_AHB_PORT7_SLAVE0, offset = 0x100) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
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

43.4.26 Security access rules for AHB peripherals

Security access rules for AHB peripherals on AHB Slave Port P7. Each peripheral can have independent security attribute.

Table 848. Security access rules for AHB peripherals. (SEC_CTRL_AHB_PORT7_SLAVE1, offset = 0x104)

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.	

Table 848. Security access rules for AHB peripherals. (SEC_CTRL_AHB_PORT7_SLAVE1, offset = 0x104) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
15:10		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

43.4.27 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 849. Security access rules for AHB peripherals. (SEC_CTRL_AHB_PORT8_SLAVE0, offset = 0x110)

Bit	Symbol	Access	Value	Description	Reset value
17:0		WO		Reserved.	0x0
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

43.4.28 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 850. Security access rules for AHB peripherals. (SEC_CTRL_AHB_PORT8_SLAVE1, offset = 0x114)

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.	
15:2		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.	
19:18		WO		Reserved.	0x0
21:20	CAN0_RULE	RW		CAN-FD.	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.	
27:22		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

43.4.29 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 851. Security access rules for AHB peripherals (SEC_CTRL_AHB_PORT9_SLAVE0, offset = 0x120)

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.	
15:2		WO		Reserved.	0x0

Table 851. Security access rules for AHB peripherals (SEC_CTRL_AHB_PORT9_SLAVE0, offset = 0x120) *…continued*

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.	
27:22		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

43.4.30 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 852. Security access rules for AHB peripherals (SEC_CTRL_AHB_PORT9_SLAVE1, offset = 0x124)

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
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

43.4.31 Security control AHB 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 853. Security control AHB (SEC_CTRL_AHB_MEM_RULE, offset = 0x130)

Bit	Symbol	Access	Value	Description	Reset value
1:0	AHB_SEC_CTRL_SECT_0_RULE	RW		Address space: 0x400A_C000 - 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
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

43.4.32 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 854. 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

43.4.33 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 855. 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

43.4.34 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 856. 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

43.4.35 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 857. 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

43.4.36 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 858. 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

43.4.37 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 859. 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

43.4.38 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 860. 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

43.4.39 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 861. 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

43.4.40 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 862. 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

43.4.41 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 863. 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

43.4.42 Security violation miscellaneous information for AHB port 0

This register provides more details on most recent security violation on AHB port 0.

Table 864. 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 - 3: Reserved.	
			4:Reserved.	
			5: SDMA0.	
			6 - 9: Reserved.	
			10: SHA-2.	

Table 864. Security violation miscellaneous information for AHB port 0 (sec_vio_misc_info0, offset = 0xE80)

Bit	Symbol	Access	Description	Reset value
			11: Reserved.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved.	undefined

43.4.43 Security violation miscellaneous information for AHB port 1

This register provides more details on most recent security violation on AHB port 1.

Table 865. 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 - 4: Reserved.	
			5: SDMA0.	
			6 - 9: Reserved.	
			10: SHA-2.	
			11: Reserved.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved.	undefined

43.4.44 Security violation miscellaneous information for AHB port 2

This register provides more details on most recent security violation on AHB port 2.

Table 866. 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.	

Table 866. Security violation miscellaneous information for AHB port 2 (sec_vio_misc_info2, offset = 0xE88)

Bit	Symbol	Access	Description	Reset value
			2 - 3: Reserved.	
			4: Reserved.	
			5: SDMA0.	
			6 - 9: Reserved.	
			10: SHA-2.	
			11: Reserved.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

43.4.45 Security violation miscellaneous information for AHB port 3

This register provides more details on most recent security violation on AHB port 3.

Table 867. 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 - 3: Reserved.	
			4: Reserved.	
			5: SDMA0.	
			6 - 9: Reserved.	
			10: SHA-2.	
			11: Reserved	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

43.4.46 Security violation miscellaneous information for AHB port 4

This register provides more details on most recent security violation on AHB port 4.

Table 868. 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

Table 868. Security violation miscellaneous information for AHB port 4 (sec_vio_misc_info4, offset = 0xE90)

Bit	Symbol	Access	Description	Reset value
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 - 3: Reserved.	
			4: Reserved.	
			5: SDMA0.	
			6 - 9: Reserved.	
			10: SHA-2.	
			11: Reserved.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved.	undefined

43.4.47 Security violation miscellaneous information for AHB port 5

This register provides more details on most recent security violation on AHB port 5.

Table 869. 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 - 3: Reserved.	
			4: Reserved.	
			5: SDMA0.	
			6 - 9: Reserved.	
			10: SHA-2.	
			11: Reserved.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

43.4.48 Security violation miscellaneous information for AHB port 6

This register provides more details on most recent security violation on AHB port 6.

Table 870. 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 - 3: Reserved.	
			4: Reserved.	
			5: SDMA0.	
			6 - 9: Reserved.	
			10: SHA-2.	

Table 870. Security violation miscellaneous information for AHB port 6 (sec_vio_misc_info6, offset = 0xE98)

Bit	Symbol	Access	Description	Reset value
			11: Reserved.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved.	undefined

43.4.49 Security violation miscellaneous information for AHB port 7

This register provides more details on most recent security violation on AHB port 7.

Table 871. 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 - 3: Reserved.	
			4: Reserved.	
			5: SDMA0.	
			6 - 9: Reserved.	
			10: SHA-2.	
			11: Reserved.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved.	undefined

43.4.50 Security violation miscellaneous information for AHB port 8

This register provides more details on most recent security violation on AHB port 8.

Table 872. 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.	

Table 872. Security violation miscellaneous information for AHB port 8 (sec_vio_misc_info7, offset = 0xEA0)

Bit	Symbol	Access	Description	Reset value
			1: CPU0 System-bus.	
			2 - 3: Reserved.	
			4: Reserved.	
			5: SDMA0.	
			6 - 9: Reserved.	
			10: SHA-2.	
			11:Reserved.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved	undefined

43.4.51 Security violation miscellaneous information for AHB port 9

This register provides more details on most recent security violation on AHB port 9.

Table 873. 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 - 3: Reserved.	
			4: Reserved.	
			5: SDMA0.	
			6 - 9: Reserved.	
			10: SHA-2.	
			11: Reserved.	
			12: SDMA1 (Secure).	
31:12		RO	Reserved.	undefined

43.4.52 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 874. 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
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
31:10	-	Reserved.	undefined

43.4.53 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 875. 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

Table 875. Secure GPIO mask for port 0 pins (SEC_GPIO_MASK0, offset = 0xF80) *…continued*

BIt	Symbol	Description	Reset value
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
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

Table 875. Secure GPIO mask for port 0 pins (SEC_GPIO_MASK0, offset = 0xF80) *…continued*

BIt	Symbol	Description	Reset value
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
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

43.4.54 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 876. 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

Table 876. Secure GPIO mask for port 1 pins (SEC_GPIO_MASK1, offset = 0xF84) *…continued*

BIt	Symbol	Description	Reset value
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
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

Table 876. Secure GPIO mask for port 1 pins (SEC_GPIO_MASK1, offset = 0xF84) *…continued*

BIt	Symbol	Description	Reset value
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

43.4.55 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 877. 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 wrItten.	
		0x1	SEC_REG_REG0 cannot be wrItten.	
		0x0		
		0x3		
3:2	SEC_GPIO_MASK1_LOCK		Secure GPIO MASK1 lock.	0x2
		0x2	SEC_GPIO_MASK1 can be wrItten.	
		0x1	SEC_REG_REG1 cannot be wrItten.	
		0x0		
		0x3		
31:4	-	-	Reserved.	undefined

43.4.56 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 878. Master secure level register (MASTER_SEC_LEVEL, offset = 0xFD0)

BIt	Symbol	Value	Description	Reset value
9:0	-		Reserved.	0x0
11:10	SDMA0		System DMA 0.	0x0
			0x3=Secure privileged. 0x2=Secure non-privileged. 0x1=non-Secure privileged. 0x0=non-Secure non-privileged.	
19:12	-		Reserved.	0x0
			0x3=Secure privileged. 0x2=Secure non-privileged. 0x1=non-Secure privileged. 0x0=non-Secure non-privileged.	

Table 878. Master secure level register (MASTER_SEC_LEVEL, offset = 0xFD0) *…continued*

BIt	Symbol	Value	Description	Reset value
21:20	HASH		Hash.	0x0
			0x3=Secure privileged. 0x2=Secure non-privileged. 0x1=non-Secure privileged. 0x0=non-Secure non-privileged.	
23:22	-		Reserved.	0x0
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.	
27:26	CANFD		CAN FD.	0x0
			0x3=Secure privileged. 0x2=Secure non-privileged. 0x1=non-Secure privileged. 0x0=non-Secure non-privileged.	
29:28	-		Reserved. Read value is undefined, only zero should be wrItten.	0x0
31:30	MASTER_SEC_LEVEL_LOCK		MASTER_SEC_LEVEL lock.	0x2
		0x2	MASTER_SEC_LEVEL_LOCK can be wrItten.	
		0x1	MASTER_SEC_LEVEL_LOCK cannot be wrItten.	
		0x0		
		0x3		

43.4.57 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 879. Master secure level anti-pole register (MASTER_SEC_ANTI_POL_REG, offset = 0xFD4)

BIt	Symbol	Value	Description	Reset value
9:0	-			0xF
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	
19:12	RESERVED		Reserved. Read value is undefined, only zero should be wrItten.	0x3
			0x0=Secure privileged 0x1=Secure non-privileged 0x2=non-Secure privileged	

Table 879. Master secure level anti-pole register (MASTER_SEC_ANTI_POL_REG, offset = 0xFD4) *…continued*

Blt	Symbol	Value	Description	Reset value
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	-		Reserved.	0x3
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	
27:26	CANFD		CAN FD. Must be equal to NOT(MASTER_SEC_LEVEL.CANFD)	0x3
			0x0=Secure privileged 0x1=Secure non-privileged 0x2=non-Secure privileged	
29:28	-		Reserved. Read value is undefined, only zero should be wrItten.	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		

43.4.58 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 880. Miscellaneous control signals for in CPU0 (CPU0_LOCK_REG, offset = 0xFEC)

Blt	Symbol	Value	Description	Reset value
1:0	LOCK_NS_VTOR		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		
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		

Table 880. Miscellaneous control signals for in CPU0 (CPU0_LOCK_REG, offset = 0xFEC) *…continued*

BIt	Symbol	Value	Description	Reset value
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		

43.4.59 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 881. 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 **MISC_CTRL_REG**	0x2
7:6	ENABLE_NS_PRIV_CHECK	Same description as in **MISC_CTRL_REG**	0x2
9:8	DISABLE_VIOLATION_ABORT	Same description as in **MISC_CTRL_REG**	0x2
11:10	DISABLE_SIMPLE_MASTER_STRICT_MODE	Same description as in **MISC_CTRL_REG**	0x2
13:12	DISABLE_SMART_MASTER_STRICT_MODE	Same description as in **MISC_CTRL_REG**	0x2
15:14	IDAU_ALL_NS	Same description as in **MISC_CTRL_REG**	0x2
31:16	-	Reserved.	undefined

43.4.60 Secure control register

This register provides more control over certain system behavior as described in individual fields.

Table 882. 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	Enable secure check for AHB matrix. 0x2: disabled. 0x1: enabled (restrictive mode) Others: Reserved	0x2
5:4	ENABLE_S_PRIV_CHECK	Enable secure privilege check for AHB matrix. 0x2: disabled. 0x1: enabled (restrictive mode) Others: Reserved	0x2
7:6	ENABLE_NS_PRIV_CHECK	Enable non-secure privilege check for AHB matrix. 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	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, DMA0, DMA1. Others: Reserved	0x2

Table 882. 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

43.4.61 Security configuration

LPC55S0x/LPC550x 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.

43.4.62 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.

LPC55S0x/LPC550x 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.

43.4.63 Authenticated debug access

The SWD Debug supports both secure and non-secure debug. The LPC55S0x/LPC550x ROM provides support to safeguard secure application code from unauthorized debug access using a Debug Authentication process.

The PFR contains fields that allow disabling of secure debug and disabling on NS debug. This allows a secure developer to 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 the 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.

43.4.64 TrustZone programming of flash

LPC55S0x/LPC550x offers two stages of flash programing. The first stage is development and deployment. At this stage, the JTAG or SWD ports are used for flash erase and programing. However, once a device is deployed, this mode of flash programing is disabled and the flash can only be programmed over the air via a secure BootROM.

43.4.65 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 LPC55S0x/LPC550x 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. LPC55S0x/LPC550x supports upward transition from Cortex-M4 to Cortex-M33.

Out of reset, the M33 CPU defaults to executing in the S state. However, the BootROM code that executes prior to executing a 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, the 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.

More details can be found in BootROM part configuration chapter. Chapter 6 “LPC55S0x/LPC550x Boot ROM”.

UM11424

Chapter 44: LPC55S0x Security features

Rev. 1.5 — 21 December 2023 User manual

44.1 How to read this chapter

The Secure Hash Algorithm (SHA), Random Number Generator (RNG), AES encryption/decryption registers, PRINCE real-time encryption/decryption, PUF, DICE, UUID, and security APIs are available on all LPC55S0x devices.

44.2 Introduction

The security system on LPC55S0x utilizes 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 162 “Security system” shows an overview of the LPC55S0x 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.

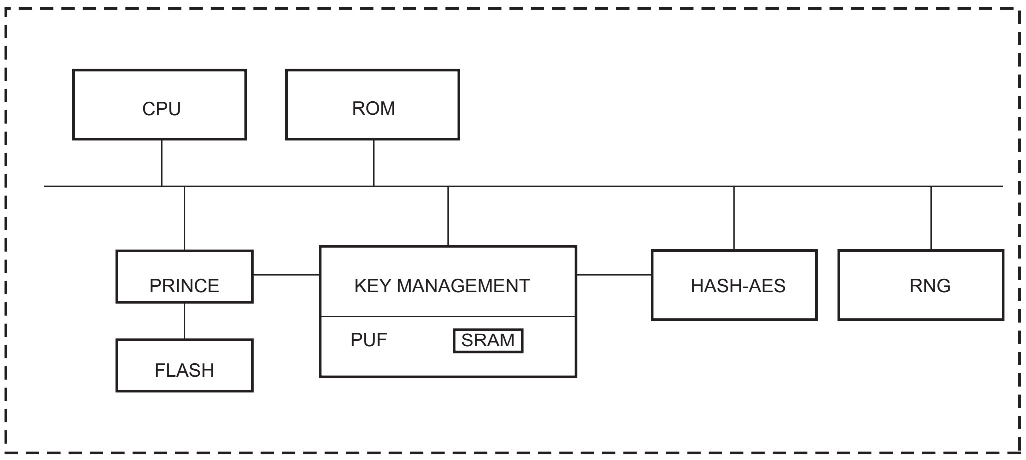


Fig 162. Security system

44.2.1 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. The LPC55S0x offers SRAM PUF where the PUF assigns a unique key to each device and exists in that device based on the unique characteristics of PUF SRAM. Figure 163 “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.

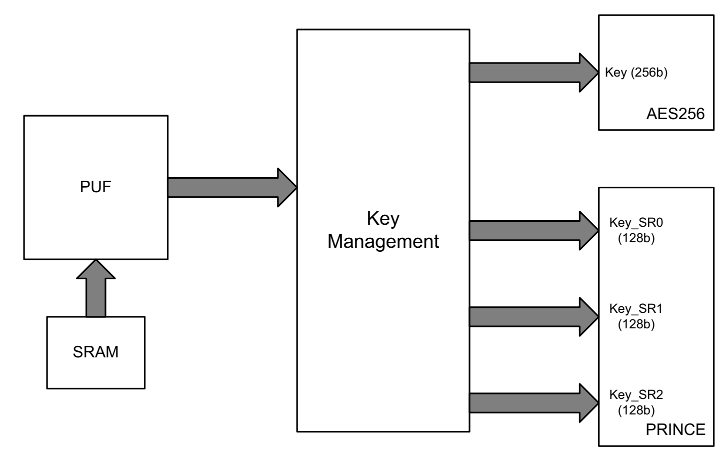


Fig 163. Key storage

44.2.1.1 PUF keys

The PUF controller generates and provides secure storage for keys and does so without storing the actual keys. This is accomplished by generating a key code based on the digital fingerprint of a device derived from SRAM which can then be used to reconstruct the keys which are then routed to the AES engine for use by software.

44.3 AES engine

The LPC55S0x devices provide an on-chip SCA (Side Channel Analysis) resistant hardware accelerator for AES encryption and decryption that conforms to the FIPS 197 standard, 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 44.12 “AES engine functional details”.

44.4 SHA

All LPC55S0x 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 whether a particular message is valid and whether or not it has been corrupted is some way.

Hashing is used for four primary purposes:

- Producing the core of a digital signature model, including certificates, and for secure updates.
- Supporting a challenge/response or to validate a message when used with a Hash-based Message Authentication Code (HMAC).
- Verifying code integrity through a secure boot model.
- Verifying that an external source of memory has not been compromised.

See Section 44.13 “HASH functional details”.

44.5 Digital signatures

A digital signature combines public/private keys such as RSA or ECC with SHA Hashing. The signature is formed in the following way:

- The message or image is hashed using SHA1, SHA2, or some other approved hashing algorithm.
- 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 the 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 model, 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 followed:

- 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.

44.6 Hash-based Message Authentication Code (HMAC)

An HMAC can be achieved on the LPC55S0x by using a pre-shared key and hashing. In this way, a message (encrypted or not) can also be used for a 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 can therefore 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 perhaps as a board-unique key shared by two devices.

44.7 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 44.15 “RNG functional details”.

44.8 UUID

The LPC55S0x stores a 128-bit IETF RFC4122 compliant non-sequential Universally Unique Identifier (UUID). It can be read from the flash PFR region at register location 0x0009_FC70 onwards.

44.9 DICE

LPC55S0x/LPC550x (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.

44.10 PRINCE real-time encryption/decryption

LPC55S0x 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 44.17.1 “Functional details”

44.11 PUF controller and key management

The PUF controller provides a secure key storage without injecting or provisioning device unique PUF root key. See Section 44.2.1.1 "PUF keys" for more details.

44.11.1 PUF controller features

The PUF controller has the following features:

- Key strength of 256-bits.
  - The PUF constructs a 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 an 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 the 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 assigns a 4-bit index value for each key while generating key codes. Keys that are assigned index value zero are output through the HW bus and are accessible to the 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.

44.11.2 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.

44.11.3 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. This step only needs to be performed 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. This is generated 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 (KC). This KC can be stored externally. This operation is performed 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. This operation must be performed every time a key is needed.

44.11.4 Key management

The LPC55S0x key management module stores an AES Key (KEY0) and three PRINCE Keys (KEY1, KEY2, KEY3). These keys are input 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 LPC55S0x 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 during deep-power down. The CTRL, CFG, KEYLOCK, KEYENABLE, KEYRESET, IDXBLK,IDXBLK_DP, SHIFT_STATUS registers are preserved and are ffnot 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.

44.11.4.1 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; if multiple keys are enabled, then no key will be enabled.
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 44.11.7.11 “Get Key” and Section 44.11.8.6 “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.

44.11.5 PUF controller register interface

Table 883 shows the registers and their addresses.

Table 883. PUF controller registers (base address = 0x4003 B000)

Name	Access	Address	Description	Reset value	Section
CTRL	R/W	0x00	PUF control register.	0x0	44.11.5.1
KEYINDEX	R/W	0x04	PUF key index register.	0x0	44.11.5.2
KEYSIZE	R/W	0x08	PUF key size register.	0x0	44.11.5.3
STAT	R	0x20	PUF status register.	0x0	44.11.5.4
ALLOW	R	0x28	PUF allow register.	0x0	44.11.5.5
KEYINPUT	W	0x40	PUF key input register.	0x0	44.11.5.6
CODEINPUT	W	0x44	PUF code input register.	0x0	44.11.5.7
CODEOUTPUT	R	0x48	PUF code output register.	0x0	44.11.5.8
KEYOUTINDEX	R	0x60	PUF key output index register.	0x0	44.11.5.9
KEYOUTPUT	R	0x64	PUF key output register.	0x0	44.11.5.10
IFSTAT	R/W1C	0xDC	PUF interface status and clear.	0x0	44.11.5.11
INTEN	RW	0x100	Interrupt enable.	0x0	44.11.5.12
INTSTAT	RO/W1C	0x104	Interrupt status	0x0	44.11.5.13
CFG	RW	0x10C	PUF config register for block bits	0x0	44.11.5.14
KEYLOCK	RW	0x200	Key lock register.	0x0	44.11.5.15
KEYENABLE	RW	0x204	Key enable register.	0x0	44.11.5.16
KEYRESET	WO	0x208	Key reset register.	0x0	44.11.5.17
IDXBLK	WO	0x20C	IDXBLK register (IDX0 - IDX15).	0x2	44.11.5.18
IDXBLK_DP	WO	0x210	IDXBLK_DP register (IDX0 - IDX15).	0x2	44.11.5.19
KEYMASK0	WO	0x214	Key mask0 register.	0x0	44.11.5.20
KEYMASK1	WO	0x218	Key mask1 register.	0x0	44.11.5.20
KEYMASK2	WO	0x21C	Key mask2 register.	0x0	44.11.5.20
KEYMASK3	WO	0x220	Key mask3 register.	0x0	44.11.5.20
IDXBLK_STATUS	R	0x254	Index block status (IDX0 - IDX15).	0x6AAAAAAA	44.11.5.21
SHIFT_STATUS	R	0x258	Shift status register.	0x0	44.11.5.22

The PUF register bits are defined in the following sections.

44.11.5.1 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 884. 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
5	-	Reserved.	0
6	GETKEY	Begin Get Key operation.	0
31:7	-	Reserved. Read value is 0, only 0 should be written.	0

44.11.5.2 PUF key index register

The PUF key index register defines the key index for the next set key operation.

Table 885. 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

44.11.5.3 PUF key size register

The PUF key size register defines the key index for the next set key operation.

Table 886. 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 44.11.7.3 “Key and code sizes”.

44.11.5.4 PUF status register

The PUF status register indicates the current status of the PUF and indicates which data is requested or available.

Table 887. 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

Table 887. PUF status register (STAT, offset = 0x20) *…continued*

Bit	Symbol	Description	Reset value
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 44.11.7.1 “Order of operations”.

44.11.5.5 PUF allow register

The PUF allow register indicates which operations are currently allowed.

Table 888. 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.

44.11.5.6 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 889. 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

44.11.5.7 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 890. 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

44.11.5.8 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 891. 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

44.11.5.9 PUF key output index register

The key index of the reconstructed key can be read using the PUF key output index register.

Table 892. 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

44.11.5.10 PUF key output register

The reconstructed key can be read using the PUF key output register.

Table 893. 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

44.11.5.11 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 894. 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.	-

44.11.5.12 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 895. 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

Table 895. PUF interrupt enable register (INTEN, offset = 0x100) *…continued*

Bit	Symbol	Description	Reset value
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

44.11.5.13 PUF interrupt status register

The PUF interrupt status register provides a view of interrupt flags that are currently enabled.

Table 896. 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

44.11.5.14 PUF Configuration register

The PUF configuration register provides configuration for block bits.

Table 897. PUF configuration register for block bits (CFG, offset 0x10C)

Bit	Symbol	Description	Reset value
0	BLOCKENROLL_SETKEY	Block enroll operation. Write 1 to set, cleared on reset.	0x0
1	BLOCKKEYOUTPUT	Block set key operation. Write 1 to set, cleared on reset.	0x0
31:2	-	Reserved. Read value is 0.	undefined

44.11.5.15 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 898. 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

44.11.5.16 Key enable register

The PUF key enable register allows user to load PUF output as secret key to a particular engine.

Table 899. 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.	

Table 899. Key enable register (KEYENABLE, offset = 0x204) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
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

44.11.5.17 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 900. 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.	
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

44.11.5.18 Index blocking register (IDX0 - IDX15)

The PUF index blocking register blocks a given index from PUF index 0-15. With the IDXn bit set, Key output from that index is not available on the APB register interface. Index blocking would be activated if relevant key fields in IDXBLK and IDXBLK_DP do not match. For example, IDX2 would only be accessible if IDX2 = 0x2 in both the IDXBLK and IDXBLK_DP registers.

Note: For each PUF index: once 0x1 is written in the IDX"n" field, the PUF index "n" is locked, and its value cannot be modified until a Power On Reset (PoR) occurs.

Table 901. Index blocking register (IDXBLK offset = 0x20C)

Bit	Symbol	Access	Value	Description	Reset value
1:0	IDX0	WO		Blocks PUF index 0.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
3:2	IDX1	WO		Blocks PUF index 1.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
5:4	IDX2	WO		Blocks PUF index 2.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
7:6	IDX3	WO		Blocks PUF index 3.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
9:8	IDX4	WO		Blocks PUF index 4.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
11:10	IDX5	WO		Blocks PUF index 5.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
13:12	IDX6	WO		Blocks PUF index 6.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
15:14	IDX7	WO		Blocks PUF index 7.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
17:16	IDX8	WO		Blocks PUF index 8.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
19:18	IDX9	WO		Blocks PUF index 9.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	

Table 901. Index blocking register (IDXBLK offset = 0x20C) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
21:20	IDX10	WO		Blocks PUF index 10.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
23:22	IDX11	WO		Blocks PUF index 11.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
25:24	IDX12	WO		Blocks PUF index 12.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
27:26	IDX13	WO		Blocks PUF index 13.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
29:28	IDX14	WO		Blocks PUF index 14.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
31:30	IDX15	WO		Blocks PUF index 15.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	

44.11.5.19 Index blocking duplicate register (IDX0 - IDX15)

This register is duplicate of IDXBLK register and provides protection against malicious attacks. Index blocking is activated if relevant key fields in IDXBLK and IDXBLK_DP do not match. For example, IDX12 is accessible if IDX12 = 0x2 in both IDXBLK and IDXBLK_DP registers.

Note: For each PUF index: once 0x1 is written in the IDX"n" field, the PUF index "n" is locked, and its value cannot be modified until a Power On Reset (PoR) occurs.

Table 902. (IDXBLK_DP, offset = 0x210)

Bit	Symbol	Access	Value	Description	Reset value
1:0	IDX0	WO		Blocks PUF index 0.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	

Table 902. (IDXBLK_DP, offset = 0x210) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
3:2	IDX1	WO		Blocks PUF index 1.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
5:4	IDX2	WO		Blocks PUF index 2.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
7:6	IDX3	WO		Blocks PUF index 3.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
9:8	IDX4	WO		Blocks PUF index 4.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
11:10	IDX5	WO		Blocks PUF index 5.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
13:12	IDX6	WO		Blocks PUF index 6.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
15:14	IDX7	WO		Blocks PUF index 7.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
17:16	IDX8	WO		Blocks PUF index 8.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
19:18	IDX9	WO		Blocks PUF index 9.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
21:20	IDX10	WO		Blocks PUF index 10.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	

Table 902. (IDXBLK_DP, offset = 0x210) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
23:22	IDX11	WO		Blocks PUF index 11.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
25:24	IDX12	WO		Blocks PUF index 12.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
27:26	IDX13	WO		Blocks PUF index 13.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
29:28	IDX14	WO		Blocks PUF index 14.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
31:30	IDX15	WO		Blocks PUF index 15.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	

44.11.5.20 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

44.11.5.21 Index block status (IDX0 - IDX15)

This register provides index block status.

Table 903. Index block status (IDXBLK_STATUS, offset 0x254)

Bit	Symbol	Access	Value	Description	Reset value
1:0	IDX0	RO		Status block index 0.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	

Table 903. Index block status (IDXBLK_STATUS, offset 0x254) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
3:2	IDX1	RO		Status block index 1.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
~~5:4~~	IDX2	RO		Status block index 2.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
7:6	IDX3	RO		Status block index 3.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
9:8	IDX4	RO		Status block index 4.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
11:10	IDX5	RO		Status block index 5.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
13:12	IDX6	RO		Status block index 6.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
15:14	IDX7	RO		Status block index 7.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
17:16	IDX8	RO		Status block index 8.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
19:18	IDX9	RO		Status block index 9.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
21:20	IDX10	RO		Status block index 10.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	

Table 903. Index block status (IDXBLK_STATUS, offset 0x254) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
23:22	IDX11	RO		Status block index 11.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
25:24	IDX12	RO		Status block index 12.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
27:26	IDX13	RO		Status block index 13.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
29:28	IDX14	RO		Status block index 14.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	
31:30	IDX15	RO		Status block index 15.	0x2
			0x2	PUF index is accessible.	
			0x1	PUF index is blocked.	
			Others	Reserved.	

44.11.5.22 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 904. (SHIFT_STATUS, offset = 0x258)

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

44.11.6 PUF SRAM Control register interface

Table 905 shows the registers and their addresses.

Table 905. PUF SRAM control registers (PUF_SRAM_CTRL) base address = 0x4003 B000)

Name	Access	Address	Description	Reset value	Section
CFG	R/W	0x300	Configuration register.	0x0	44.11.6.1
STATUS	WO	0x304	Status register.	0x0	44.11.6.2
INT_CLR_ENABLE	WO	0x3D8	Interrupt enable clear register.	0x0	44.11.6.3
INT_SET_ENABLE	WO	0x3DC	Interrupt enable set register.	0x0	44.11.6.4
INT_STATUS	R	0x3E0	Interrupt status register.	0x0	44.11.6.5
INT_ENABLE	R	0x3E4	Interrupt enable register.	0x0	44.11.6.6
INT_CLR_STATUS	WO	0x3E8	Interrupt status clear register.	0x0	44.11.6.7
INT_SET_STATUS	WO	0x3EC	Interrupt status set register.	0x0	44.11.6.8

The SRAM PUF register bits are defined in the following sections.

44.11.6.1 Configuration register

The PUF SRAM configuration bit is used to perform configuration functions.

Table 906. PUF SRAM Configuration register (CFG, offset 0x300)

Bit	Symbol	Value	Description	Reset value
1:0	ENABLE		PUF SRAM Controller activation.	0
		0	Disabled.	
		1	Enabled.	
2	CKGATING		PUF SRAM Clock Gating control.	0
		0	Disabled.	
		1	Enabled.	
31:3	-		Reserved. Read value is 0, only 0 should be written.	0

44.11.6.2 Status register

The status register.

Table 907. PUF SRAM status register (STATUS, offset = 0x304)

Bit	Symbol	Value	Description	Reset value
0	READY		PUF SRAM Controller state.	0
		0	Not ready. PUF cannot access PUF SRAM.	
		1	Ready. PUF can access PUF SRAM.	
31:1	-		Reserved. Read value is 0, only 0 should be written.	0

44.11.6.3 Interrupt enable clear register

The Interrupt enable clear register....

Table 908. Interrupt enable clear register (INT_CLR_ENABLE, offset = 0x3D8

Bit	Symbol	Value	Description	Reset value
0	READY		Ready interrupt enable clear.	0
		0	No effect.	
		1	Clears the READY bit field in register INT_ENABLE. Automatically reset by the Hardware.	
1	APB_ERR		APB_ERR Interrupt enable clear.	0
		0	No effect.	
		1	Clears the APB_ERR bit field in register INT_ENABLE. Automatically reset by the Hardware.	
31:2	-		Reserved. Read value is 0, only 0 should be written.	0

44.11.6.4 Interrupt enable set register

The interrupt enable set register...

Table 909. PUF Interrupt enable set register (INT_SET_ENABLE, offset = 0x3DC)

Bit	Symbol	Value	Description	Reset value
0	READY		Ready interrupt enable set.	0
		0	No effect.	
		1	Sets the READY bit field in register INT_ENABLE. Automatically reset by the Hardware.	
1	APB_ERR		APB_ERR Interrupt enable set.	0
		0	No effect.	
		1	Sets the APB_ERR bit field in register INT_ENABLE. Automatically reset by the Hardware.	
31:2	-		Reserved. Read value is 0, only 0 should be written.	0

44.11.6.5 Interrupt status register

The interrupt status register.

Table 910. PUF interrupt status register (INT_STATUS, offset = 0x3E0)

Bit	Symbol	Value	Description	Reset value
0	READY		Ready interrupt status. The ready interrupt status, which is based on the value of the READY bit in the NT_ENABLE register, is set at the end of the SRAM initialization phase to indicate that the appropriate SRAM is available for PUF utilization. An interrupt is generated only when both this bit and the READY bit in INT_ENABLE register are set.	0
		0	Not pending.	
		1	Pending.	
1	APB_ERR		APB_ERR Interrupt Status. Set when access to a register is denied based on the access security rules determined by pprot,pprot_mask and pprot_match.	0

Table 910. PUF interrupt status register (INT_STATUS, offset = 0x3E0)

Bit	Symbol	Value	Description	Reset value
		0	Not pending.	
		1	Pending.	
31:2	-		Reserved. Read value is 0, only 0 should be written.	0

44.11.6.6 Interrupt enable register

The interrupt enable register.

Table 911. PUF interrupt enable register (INT_ENABLE, offset = 0x3E4)

Bit	Symbol	Value	Description	Reset value
0	READY		Ready interrupt enable.	0
		0	Disabled.	
		1	Enabled.	
1	APB_ERR		APB_ERR interrupt enable.	0
		0	Disabled.	
		1	Enabled.	
31:2	-		Reserved. Read value is 0, only 0 should be written.	0

44.11.6.7 Interrupt status clear register

The interrupt status clear register.

Table 912. PUF interrupt status clear register (INT_CLR_STATUS, offset = 0x3E8)

Bit	Symbol	Value	Description	Reset value
0	READY		Ready interrupt status clear.	0
		0	No effect.	
		1	Enabled.	
1	APB_ERR		APB_ERR interrupt status clear.	0
		0	No effect.	
		1	Clears the APB_ERR bit field in register INT_STATUS. Automatically reset by the hardware.	
31:2	-		Reserved. Read value is 0, only 0 should be written.	0

44.11.6.8 Interrupt status set register

The interrupt status set register.

Table 913. PUF interrupt status set register (INT_SET_STATUS, offset = 0x3EC)

Bit	Symbol	Value	Description	Reset value
0	READY		READY Interrupt Status set.	0
		0	No effect.	
		1	Sets the READY bit field in register INT_STATUS. Automatically reset by the Hardware.	
1	APB_ERR		APB_ERR Interrupt Status Set.	0

Table 913. PUF interrupt status set register (INT_SET_STATUS, offset = 0x3EC)

Bit	Symbol	Value	Description	Reset value
		0	No effect.	
		1	Sets the APB_ERR bit field in register INT_STATUS. Automatically reset by the Hardware.	
31:2	-		Reserved. Read value is 0, only 0 should be written.	0

44.11.7 Using PUF

This section describes steps for setting up and usage of PUF block.

44.11.7.1 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 164. 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 initial start operation, 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.

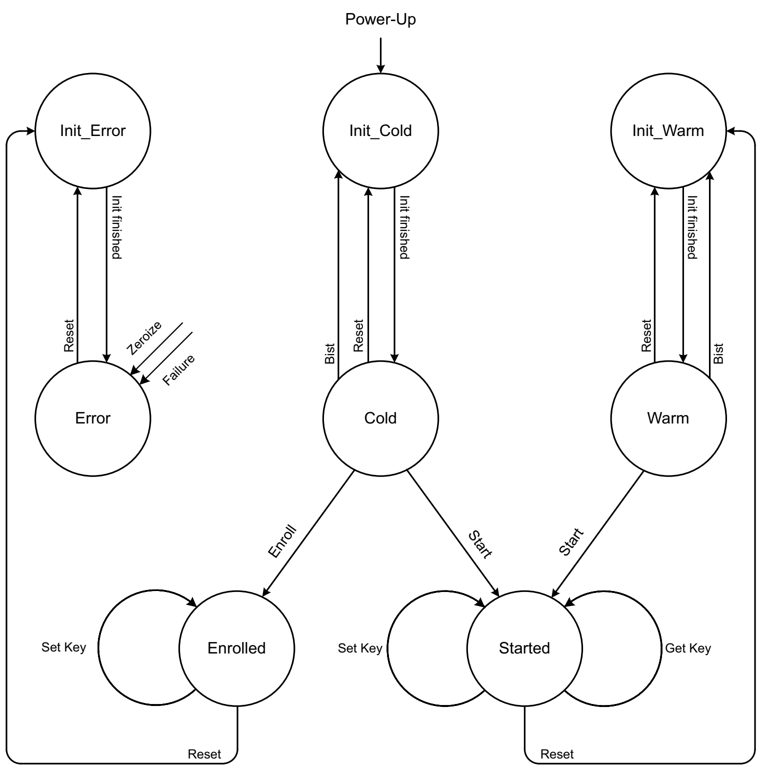


Fig 164. Possible flows, states, and actions

In case of a ZEROIZE operation through the control register or a failure (for example, a wrong activation code is provided), the PUF controller goes into an 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 914, 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 914. 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

44.11.7.2 Activation code size

The size of the Activation Code (AC size) is:

```
AC size = 9536 bits (1192 bytes)
```

44.11.7.3 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 and so 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 915 specifies the KEYSIZE values to use for the supported key sizes, and the size of the related PUF-generated KC.

Table 915. 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

Table 915. Coding of KEYSIZE *…continued*

KEYSIZE (5:0)	Value	Key size (bits)	KC size (bits)	KC size (bytes)
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

44.11.7.4 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. Next, the Set Key operation is started, the key is passed to the PUF, and 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.

44.11.7.5 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 916. The unused bits are always 0.

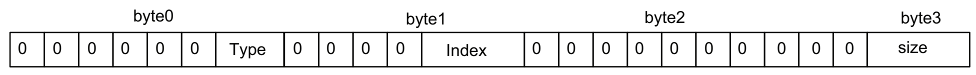


Fig 165. KC header format

Table 916. 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.

44.11.7.6 Key byte order on the APB interface

The first word contains the first bytes and the second word the next bytes. Figure 166 shows the byte 16 order within a word.

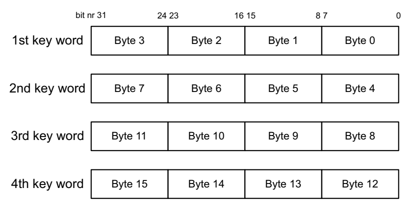


Fig 166. Key byte order on the Key interface for 128-bit key

44.11.7.7 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 the 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.

44.11.7.8 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 is moved to the Error state. See Figure 164 and Section 44.11.7.13 “Error response”.

44.11.7.9 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.

44.11.7.10 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.

44.11.7.11 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 917. See Section 44.11.7.4 “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 164 and Section 44.11.7.13 “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 917. 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.

44.11.7.12 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 164. No new operations can be performed until the device is repowered.

44.11.7.13 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.

44.11.7.14 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.

44.11.8 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 44.11.7 “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 918 defines the parameters of the functions. Table 919 defines the data access functions; these must be supplied by the system.

Key formats are defined in section Section 44.11.7.6 “Key byte order on the APB interface”. It is assumed that the data and key are stored in memory in this format.

Table 918. 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 919. 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.

44.11.8.1 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
}
```

44.11.8.2 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
}
```

44.11.8.3 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
}
```

44.11.8.4 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
}
```

44.11.8.5 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
}
```

44.11.8.6 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
}
```

44.11.8.7 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
}
```

44.12 AES engine functional details

The AES engine supports 128-bit, 192-bit, or 256-bit keys for encryption and decryption operations.

44.12.1 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.
- AES engine provides side-channel analysis resistance thanks to special SCA protection techniques implemented in the engine.
- Security against Side Channel Analysis (power & Electro Magnetic traces) using masking techniques to protect against DPA (Differential Power Analysis), CPA (Correlation Power Analysis) and template 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.

44.12.2 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 44.14.3 “Status register”.
- For AES registers please refer to SHA-AES register description Section 44.16 “Register description”.
- AES block shares same base address as SHA block.

44.12.3 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.

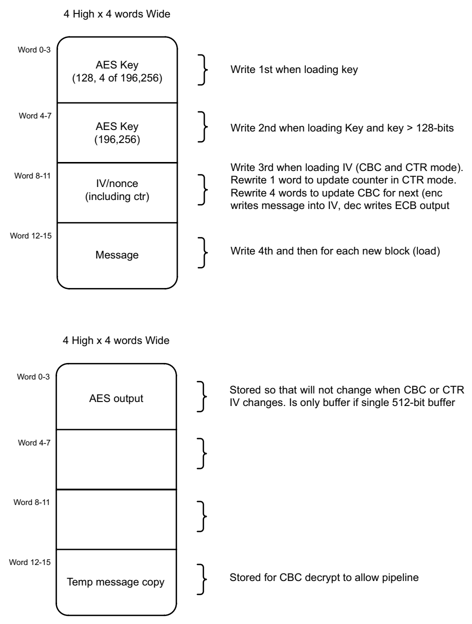

4 High x 4 words Wide
Word 0-3
AES Key (128, 4 of 196,256)
Write 1st when loading key
Word 4-7
AES Key (196,256)
Write 2nd when loading Key and key > 128-bits
Word 8-11
IV/nonce (including ctr)
Write 3rd when loading IV (CBC and CTR mode). Rewrite 1 word to update counter in CTR mode. Rewrite 4 words to update CBC for next (enc writes message into IV, dec writes ECB output
Word 12-15
Message
Write 4th and then for each new block (load)
4 High x 4 words Wide
Word 0-3
AES output
Stored so that will not change when CBC or CTR IV changes. Is only buffer if single 512-bit buffer
Word 4-7
Word 8-11
Word 12-15
Temp message copy
Stored for CBC decrypt to allow pipeline

Fig 167. 2x512-bit buffers are used for AES

44.12.4 Using AES engine

1. When starting a new operation, write the CTRL register NEW bit to initialize.
2. Read a 32-bit entropy seed from TRNG and write to the PRNG_SEED register.
3. 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 44.2.1 “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.

4. 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.
5. 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.
6. 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
7. 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.
8. 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.

44.12.5 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.

44.13 HASH functional details

44.13.1 Features

- Performs SHA-1 and SHA-2(256) based hashing.
- Used with HMAC to support a challenge/response or to validate a message.

44.13.2 Basic configuration

- The priority for the SHA engine can be set in PRI_SHA bit in AHBMATPRIO register. See Table 41.
- 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.

44.13.3 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.

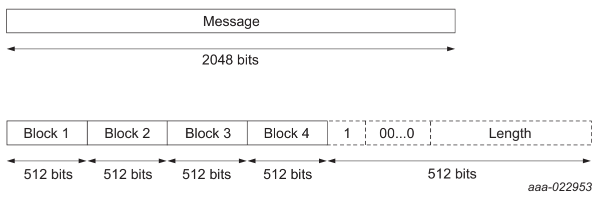


Fig 168. SHA input data block

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.

Digest registers are writable to allow interleaved Hash operations, such as multiple hash context or to interleave hashing with the AES operation. During a context switch, the software can read the running hash and store it in the system memory where the necessary AES or other hashing is performed. After completion, a context restore writes the running hash to the DIGESTn registers where the hashing continues from the point where it left off. The NEW_HASH bit in the control register can then be set to START.

44.13.4 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.

44.14 HASH-AES Register description

Table 920. Register overview: (HASH-AES, base address = 0x400A 4000)

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	Control register.	0x0	44.14.2
STATUS	R/W1C	0x004	Status register.	0x0	44.14.3
INTENSET	R/W	0x008	Interrupt enable register.	0x0	44.14.4
INTENCLR	W1C	0x00C	Interrupt clear register.	-	44.14.5
MEMCTRL	R/W	0x010	Memory control register.	0x0	44.14.6
MEMADDR	R/W	0x014	Memory address register.	0x0	44.14.7
INDATA	WO	0x020	Input data register.	0x0	44.14.8
ALIAS0	WO	0x024	Alias0 register.	-	44.14.8
ALIAS1	WO	0x28	Alias1 register.	-	44.14.8
ALIAS2	WO	0x2C	Alias2 register.	-	44.14.8
ALIAS3	WO	0x30	Alias3 register.	-	44.14.8
ALIAS4	WO	0x34	Alias4 register.	-	44.14.8
ALIAS5	WO	0x38	Alias5 register.	-	44.14.8
ALIAS6	WO	0x3C	Alias6 register.	-	44.14.8
DIGEST0/OUTDATA0	R	0x040	Digest 0 register.	0x0	44.14.9
DIGEST1/OUTDATA	R	0x44	Digest 1 register.	0x0	44.14.9
DIGEST2/OUTDATA0	R	0x48	Digest 2 register.	0x0	44.14.9
DIGEST3/OUTDATA0	R	0x4C	Digest 3 register.	0x0	44.14.9
DIGEST4/OUTDATA0	R	0x50	Digest 4 register.	0x0	44.14.9
DIGEST5/OUTDATA0	R	0x54	Digest 5 register.	0x0	44.14.9
DIGEST6/OUTDATA0	R	0x58	Digest 6 register.	0x0	44.14.9
DIGEST7/OUTDATA0	R	0x5C	Digest 7 register.	0x0	44.14.9
CRYPTCFG	RW	0x80	CRYPTCFG.	0x0	44.14.10
CONFIG	RO	0x84	CONFIG.	0x1CB	44.14.11
LOCK	RW	0x8C	LOCK.	0x0	44.14.12
MASK0	WO	0x90	MASK0.	0x0	44.14.13
MASK1	WO	0x94	MASK1.	0x0	44.14.13
MASK2	WO	0x98	MASK2.	0x0	44.14.13
MASK3	WO	0x9C	MASK3.	0x0	44.14.13
RELOAD0	W	0xA0	RELOAD0	0x0	44.14.14
RELOAD1	W	0xA4	RELOAD1	0x0	44.14.14
RELOAD2	W	0xA8	RELOAD2	0x0	44.14.14
RELOAD3	W	0xAC	RELOAD3	0x0	44.14.14
RELOAD4	W	0xB0	RELOAD4	0x0	44.14.14
RELOAD5	W	0xB4	RELOAD5	0x0	44.14.14
RELOAD6	W	0xB8	RELOAD6	0x0	44.14.14

Table 920. Register overview: (HASH-AES, base address = 0x400A 4000) *…continued*

Name	Access	Offset	Description	Reset value	Section
RELOAD7	W	0xBC	RELOAD7	0x0	44.14.14
PRNG_SEED	WO	0xD0	Provides seed input for random number generator.	0x0	44.14.15
PRNG_OUT	RO	0xD8	Provides random number.	0x0	44.14.16

44.14.1 Usage

The following section describes the programming sequence for the RNG module and relevant system settings for RNG use-scenarios.

44.14.2 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 921. 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	Reserved	
		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
5	Reload		If 1, allows the SHA RELOAD registers to be used. This is used to save a partial Hash Digest (e.g. when need to run AES) and then reload it later for continuation.	0x0
7:6	-	-	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.	

Table 921. Control register (CTRL, offset = 0x000) *…continued*

Bit	Symbol	Value	Description	Reset value
		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
13	AESFLUSH		Flushes the AES engine registers. This bit self clears.	0
		0	Do not flush the AES engine registers.	
		1	Flush the AES engine registers.	
31:14	-	-	Reserved.	-

44.14.3 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 922. 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).	

Table 922. Status register (STATUS, offset = 0x4) *…continued*

Bit	Symbol	Access	Value	Description	Reset value
3	FAULT	RO		Indicates if an AES or PRNG fault has occurred.	0x0
			0	No AES or PRNG fault has occurred.	
			1	An AES or PRNG fault has occurred.	
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.	
7:6		WO		Reserved.	undefined
8	AESFAULT	RO		AES fault status.	0x0
			0	No AES fault has occurred.	
			1	An AES fault has occurred.	
9	PRNGFAULT	RO		PRNG fault status.	0x0
			0	No PRNG fault has occurred.	
			1	A PRNG fault has occurred.	
31:10	-	-	-	Reserved.	undefined

44.14.4 Interrupt enable register

The Interrupt enable register is used to enable interrupt sources that cause processor interrupts.

Table 923. 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.	
3	FAULT		Indicates if should interrupt on an AES or PRNG fault as indicated in the STATUS register.	0
		0	No interrupt on an AES or PRNG fault.	
		1	Interrupt on an AES or PRNG fault.	
31:4	-	-	Reserved.	

44.14.5 Interrupt clear register

The Interrupt clear register is used to clear the interrupt mask enabled by the INTENSET register.

Table 924. 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
3	FAULT	Writing a 1 clears the interrupt enabled by the INTENSET register.	0
31:4	-	Reserved.	-

44.14.6 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 925. 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.	
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.	

44.14.7 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 LPC55S0x device used and must be word aligned.

Table 926. 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

44.14.8 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 927. 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 928. 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 929. 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 930. 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 931. 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 932. 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 933. 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 934. Alias 6 register (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

44.14.9 DIGEST (or OUTDATA) registers

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 935. DIGEST 0 register (DIGEST0, offset = 0x040)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 936. DIGEST 1 register (DIGEST1, offset = 0x044)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 937. DIGEST 2 register (DIGEST2, offset = 0x048)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 938. DIGEST 3 register (DIGEST3, offset = 0x04C)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 939. DIGEST 4 register (DIGEST4, offset = 0x050)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 940. DIGEST 5 register (DIGEST5, offset = 0x054)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 941. DIGEST 6 register (DIGEST6, offset = 0x058)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

Table 942. DIGEST 7 register (DIGEST7, offset = 0x05C)

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the digest.	0

44.14.10 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 943. 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 44.12.3 “General description”.	
		2	CTR mode. See Section 44.12.3 “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
31:17	-	-	Reserved.	-

44.14.11 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 944. 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	-	RO	Reserved.	
31:12	-	-	Reserved.	

44.14.12 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 44.11.5.17 “Key reset register”.

Table 945. LOCK register (LOCK, offset = 0x80C)

Bit	Symbol	Value	Description	Reset value
1:0	SECLOCK	RW	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 security level it is locked at. 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 0xA75 to change lock state.	
		A75	Pattern needed to change bits 1:0.	
31:16	-	-	Reserved.	

44.14.13 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 946. MASK registers (MASK[0:3], offset [0x090:0x9C])

Bit	Symbol	Value	Description	Reset value
31:0	MASK	WO	A random word.	0x0

44.14.14 Reload registers

The WO digest-reload registers may be written with a saved Hash digest, to allow continuation from where left off. These registers may only be written if the Reload field in CTRL is 1. If SHA1, only the first 5 are used.

Table 947. RELOAD registers (RELOAD[0:7], offset [0x0A0:0xBC])

Bit	Symbol	Value	Description	Reset value
31:0	DIGEST	WO	SHA Digest word to reload.	0x0

44.14.15 PRNG seed

Provides the seed and access to a randomly generated number.

Table 948. PRNG_SEED random input value used as an entropy source (PRNG_SEED, offset 0xD0)

Bit	Symbol	Value	Description	Reset value
31:0	PRNG_SEED	WO	This input offers a 32-bit random seed for a PRNG block that is used for masking in AES operations. This register is written at the start of an AES operation and should be updated periodically. Updating it at the start of every new AES operation is recommended to provide enough entropy. TRNG can be used as source for this value.	0x0

44.14.16 PRNG output

Provides the output for the random number seed.

Table 949. PRNG_OUT software-accessable random output value (PRNG_OUT, offset 0xD8)

Bit	Symbol	Value	Description	Reset value
31:0	PRNG_OUT	RO	This register provides a 32bit pseudo random number. This register is not accessible when the AES engine is in use.	0x0

44.14.17 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 “LPC55S0x/LPC550x 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.

44.14.17.1 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.

44.14.17.1.1 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.

44.14.17.1.2 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.

44.14.17.1.3 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.

44.14.17.2 Initialization

To setup the SHA engine:

1. Take the SHA engine out of reset mode using the HASH0_RST bit, see Section 4.5.8 "Peripheral reset control 2" and enable the clock to SHA peripheral using the HASH0 bit in the AHBCLKCTRL2 register, see Section 4.5.18 "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.

44.14.17.3 Interrupt Service Routine (ISR)

44.14.17.3.1 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.

44.14.17.3.2 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 44.14.17.3.1 “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.

44.14.17.3.3 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.

44.15 RNG functional details

This RNG module is a true random number generator based on two main sources of entropy:

- Phase noise of unprecise 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 Generator (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.

44.15.1 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).

44.15.2 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.

44.15.3 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.

44.15.4 Entropy accumulation

Entropy accumulation is linear and is estimated by hardware.

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 44.15.5 “Initialization” and Section 44.15.6 “Normal Usage”. The quality of initial entropy is based on states of 404 flip-flops which should be random.

44.15.5 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. Activate CHI computing with ONLINE_TEST_CFG.ACTIVATE = 1, with default setting for the RNG clock selection (COUNTER_CFG.CLOCK_SEL = 0). This means CHI computing is done on all clocks simultaneously. This will start accumulating linear entropy.

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 > 7, 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 <= 7, initialization is now complete. After completion of initialization, follow the below steps to read Random number.

44.15.6 Normal Usage

Here are usual steps for setting up the entropy:

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 7.
5. Go to step 2 and read new random number.

44.15.7 Checking the state of initial entropy

When checking initial entropy, for total failure or low quality, note that there is no hardware self-checking mechanism. However, entropy can be saved in the non-volatile memory before a power down then restored after power-up using dedicated registers. (See Section 44.16.6 “Entropy inject register” for more information). This gives control over initial entropy.
Additionally, some level of software workaround can be implemented:

- SW can use the same procedure described above to read the initial number in order to ensure a minimum entropy accumulation after power-up.
- SW may store initial values in non-volatile memory and compute some statistics.

44.15.8 Checking run-time entropy

Run-time entropy quality is judged as follows:

- 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 low quality of entropy source.

In addition, injection of additional external entropy is possible as an option, using a dedicated register (see Section ”Entropy inject register”).

44.16 Register description

Table 950 shows the registers and their addresses.

Table 950. 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]	44.16.1
COUNTER_VAL	R	0x8	Counter validation register.	0	44.16.2
COUNTER_CFG	RW	0xC	Counter configuration register.	0x2	44.16.3
ONLINE_TEST_CFG	RW	0x10	Online test configuration.	0	44.16.4
ONLINE_TEST_VAL	R	0x14	Online test validation.	0	44.16.5
ENTROPY_INJECT	RW	0x18	Entropy inject register.	0000	44.16.6
MODULEID	R	0xFFC	Module ID.	0xA0B84200	44.16.7

[1] This register provides random number after reset.

44.16.1 Random number register

This register holds random number generated by the engine.

Table 951. Random number register (RANDOM_NUMBER, offset = 0X0)

Bit	Symbol	Description	Reset value
31:0	RANDOM_NUMBER	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]

[1] This register provides random number after reset.

44.16.2 Counter validation register

This register provides RNG relevant information for evaluation and certification purposes.

Table 952. 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
21:8	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 7), 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: Entropy accumulation is linear.	0
31:22	-	Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.	0

44.16.3 COUNTER configuration register

This register provides RNG relevant settings for evaluation and certification purposes. Please refer to Section 44.15.2 “Certification”.

Table 953. 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.	0x2
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 Note: The recommendation is to use CHI Computing on all clocks simultaneously, that is use setting ‘000’.	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.	

44.16.4 Online test configuration register

Table 954. 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	Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.	0

44.16.5 Online test validation register

Table 955. 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

44.16.6 Entropy inject register

Table 956. Entropy inject register (ENTROPY_INJECT, offset = 0X18)

Bit	Symbol	Description	Reset value
31:0	ENTROPY	Use this register to inject or restore entropy 32 bits at a time. TRNG input clocks should be activated before writing to this register. Injection can be useful for contributing entropy from external sources such as LSBs of an ADC or temperature sensor. The Entropy from random number register can be stored in device nonvolatile memory before putting device in power-down mode. This saved entropy can be read in entropy injection register after device powers up. This way entropy can be restored back to RNG module after power-up by injecting entropy using entropy injection register. The recommendation is to inject 32 bit at a time through this entropy injection register. The maximum capacity of restoration is about 176 bits. User can inject less than 32 bit word (set unused bits to 0). Injection does not degrade overall performance due to the fact that some internal PRNGs are excluded from this external action.	0000

44.16.7 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 957. Module identification register (MODULEID, 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.	4
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

44.17 PRINCE real-time encryption or decryption details

44.17.1 Functional details

The LPC55S0x supports three regions for encryption and decryption, referred to as crypto regions. See Figure 169 below. Each crypto region resides at a 256 kB address boundary within the flash. All three regions have a start address of 0x0 and all three regions are overlapped. 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 44.11.4 “Key management” for more details.

In Figure 169, 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.

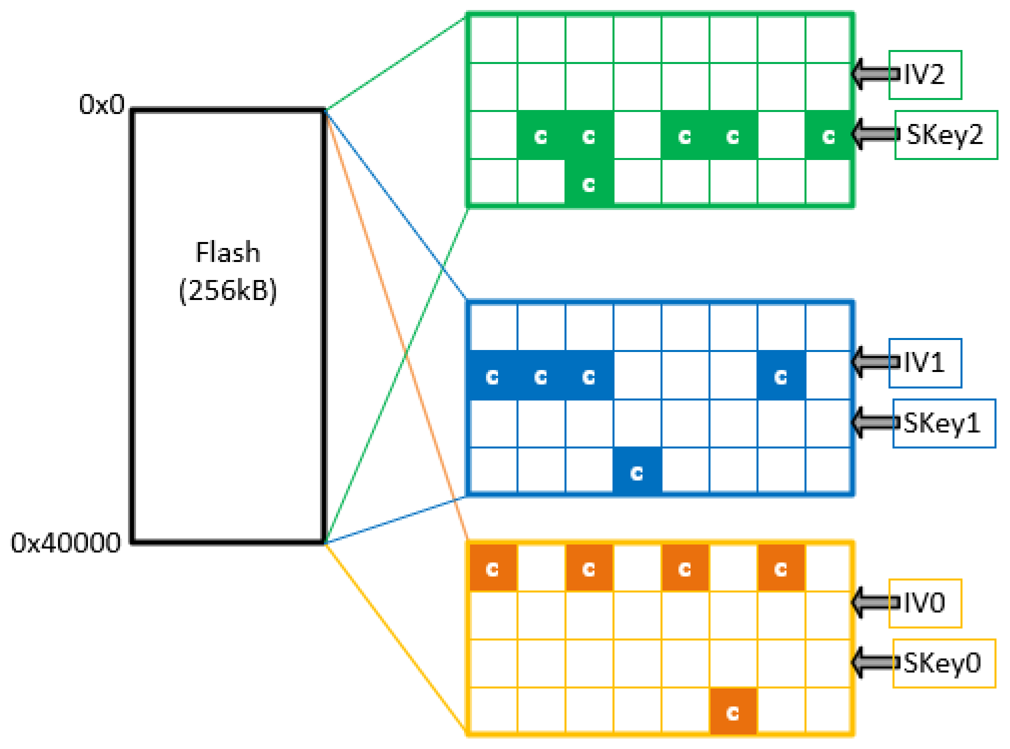


Fig 169. Conceptual diagram of flash crypto regions

PRINCE registers are retained during deep-sleep and power-down but not retained during deep-power down.

44.17.2 Usage notes

During flash programming, the PRINCE control logic monitors the address contained in the STARTA register of the flash controller to determine when data writes to DATAW0-DATAW3 registers of the flash controller, 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 the DATAW0-DATAW3 registers, because they are used also 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.

This bit self-clears at the end of each flash program operation. Reading of encrypted flash regions is disabled when ENC_ENABLE.EN is set. When set, reads of the PRINCE-encrypted regions will return invalid data, and cause the error status bit be set in the ERR register.

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 44.11.4 “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.

Remark: The ENC_ENABLE register enables encryption of write data 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, decryption is enabled when ENC_ENABLE.EN=0 and the SR_ENABLE bit of the corresponding sub region is set.

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.

44.18 PRINCE register descriptions

44.18.1 PRINCE memory map

Table 958 shows the registers and their addresses.

Table 958. Register overview: (PRINCE, base address = 5003_5000h)

Name	Access	Offset	Description	Reset value	Section
ENC_ENABLE	RW	0x0	Encryption enable register.		44.18.2
MASK_LSB	WO	0x4	Data mask register, 32 Least Significant Bits.	0000_0000h	44.18.3
MASK_MSB	WO	0x8	Data mask register, 32 Most Significant Bits.	0000_0000h	44.18.4
LOCK	RW	0xC	Lock register.		44.18.5
IV_LSB0	WO	0x10	Initial vector register for region 0, Least Significant Bits.	0000_0000h	44.18.6
IV_MSB0	WO	0x14	Initial vector register for region 0, Most Significant Bits.	0000_0000h	44.18.7
BASE_ADDR0	RW	0x18	Base Address for region 0 register.	0000_0000h	44.18.8
SR_ENABLE0	RW	0x1C	Sub-region enable register for region 0	0000_0000h	44.18.9
IV_LSB1	WO	0x20	Initial vector register for region 1, Least Significant Bits.	0000_0000h	44.18.10
IV_MSB1	WO	0x24	Initial vector register for region 1, Most Significant Bits.	0000_0000h	44.18.11
BASE_ADDR1	RW	0x28	Base Address for region 1 register.	0004_0000h	44.18.12
SR_ENABLE1	RW	0x2C	Sub-region enable register for region 1.	0000_0000h	44.18.13
IV_LSB2	WO	0x30	Initial vector register for region 2, Least Significant Bits.	0000_0000h	44.18.14
IV_MSB2	WO	0x34	Initial vector register for region 2, Most Significant Bits.	0000_0000h	44.18.15

Table 958. Register overview: (PRINCE, base address = 5003_5000h) *…continued*

Name	Access	Offset	Description	Reset value	Section
BASE_ADDR2	RW	0x38	Base Address for region 2 register.	0008_0000h	44.18.16
SR_ENABLE2	RW	0x3C	Sub-region enable for region 2 register.	0000_0000h	44.18.17
ERR	RW	0x90	Error status register.	undefined	44.18.18

44.18.2 Encryption enable register (ENC_ENABLE)

This register controls whether the PRINCE encryption functionality is enabled.

Table 959. Encryption enable register (ENC_ENABLE, offset = 0x0)

Bit	Symbol	Access	Value	Description	Reset value
0	EN	RW		Enables PRINCE encryption for flash programming. When set, reads of PRINCE-encrypted regions will return bad data and cause the error status bit be set in the ERR register. The EN bit self-clears when a flash controller program or erase operation is performed.	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. Reading of PRINCE-encrypted flash regions is disabled.	
31:1		WO		Reserved. Read value is undefined, only zero should be written.	undefined

44.18.3 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 960. 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

44.18.4 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 961. 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

44.18.5 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 962. 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

44.18.6 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 963. 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

44.18.7 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 964. 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

44.18.8 Base Address for region 0 register (BASE_ADDR0)

This register provides the base address for region 0, register 0.

Table 965. 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.	

44.18.9 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 966. 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

44.18.10 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 967. 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

44.18.11 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 968. 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

44.18.12 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 969. 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.	

44.18.13 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 970. 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

44.18.14 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 971. 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

44.18.15 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 972. 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

44.18.16 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 973. 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.	

44.18.17 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 974. 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 the PRINCE.	0x0

44.18.18 Error status register

This register is used for determining error status.

Table 975. Error status register (ERR, offset 0x90)

Bit	Symbol	Access	Value	Description	Reset value
0	ERRSTAT	RW		PRINCE Error Status. This bit is write-1 to clear.	0x0
			0	No PRINCE error.	
			1	Error. A read of a PRINCE-encrypted region was attempted while ENC_ENABLE.EN=1.	
31:1	-	-		Reserved.	0x0

UM11424

Chapter 45: LPC55S0x/LPC550x CASPER Peripheral Access Layer

Rev. 1.5 — 21 December 2023 **User manual**

45.1 How to read this chapter

The CASPER peripheral is available on all LPC55S0x/LPC550x devices. This 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, greater than four times the usual performance can often be achieved.

Supported cryptographic functions are implemented in the SDK (Software Development Kit) and the mbed TLS examples utilize the CASPER peripheral for computations.

45.2 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 976 for a performance comparison for the various functions:

Table 976.

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

45.3 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 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:

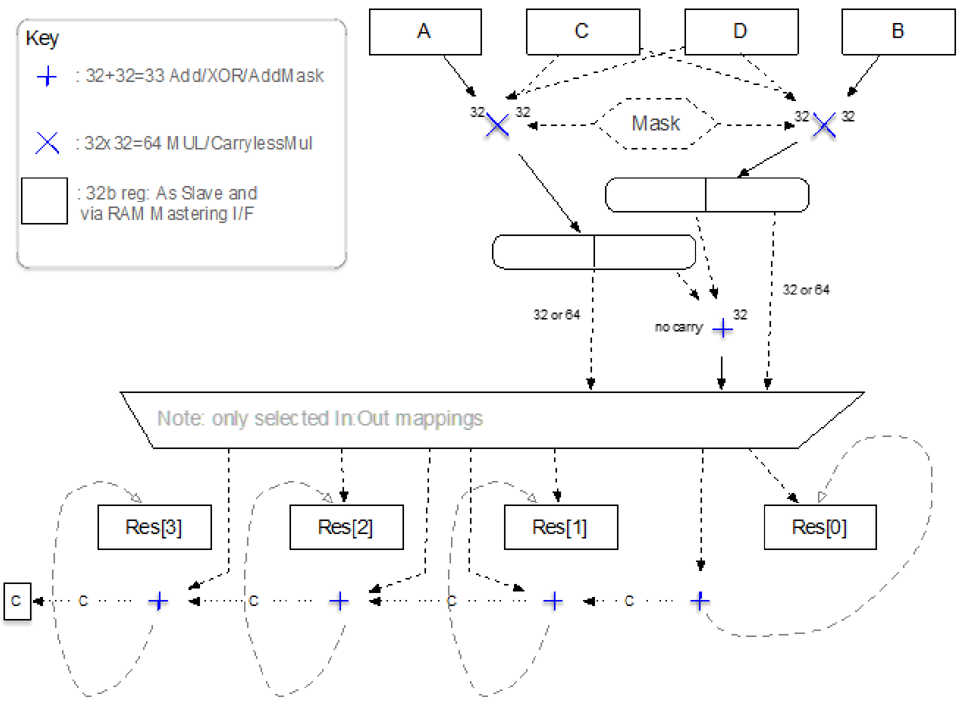

Key
: 32+32=33 Add/XOR/AddMask
: 32x32=64 MUL/CarrylessMul
: 32b reg: As Slave and via RAM Mastering I/F
A
C
D
B
32
32
Mask
32
32
32 or 64
no carry
32
32 or 64
Note: only selected In:Out mappings
Res[3]
Res[2]
Res[1]
Res[0]
C
c
c
c
c

Fig 170. 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:

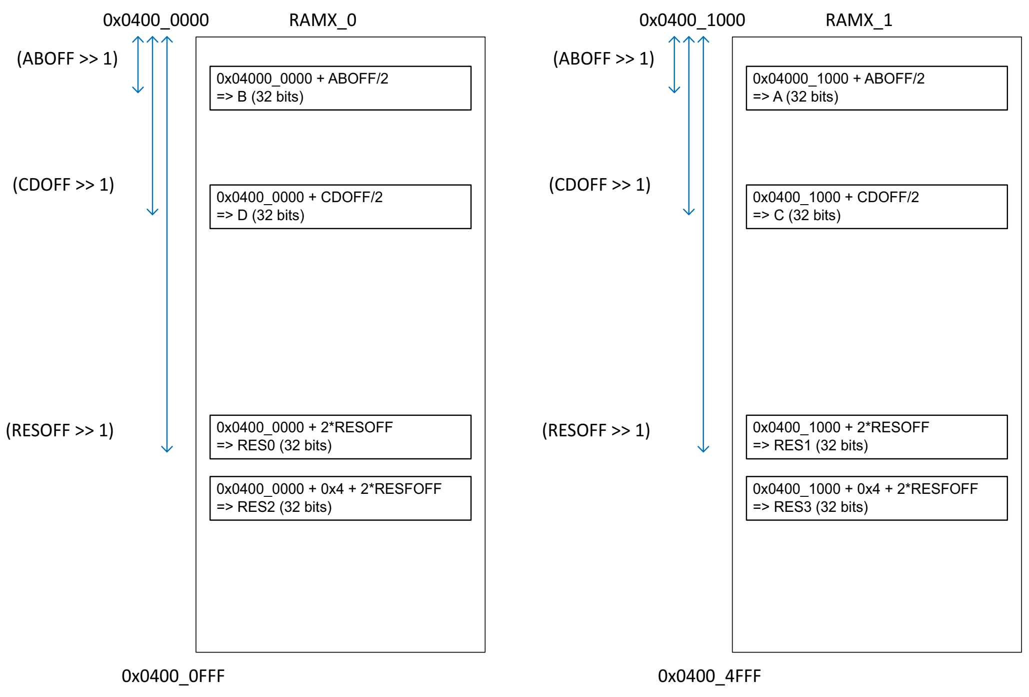
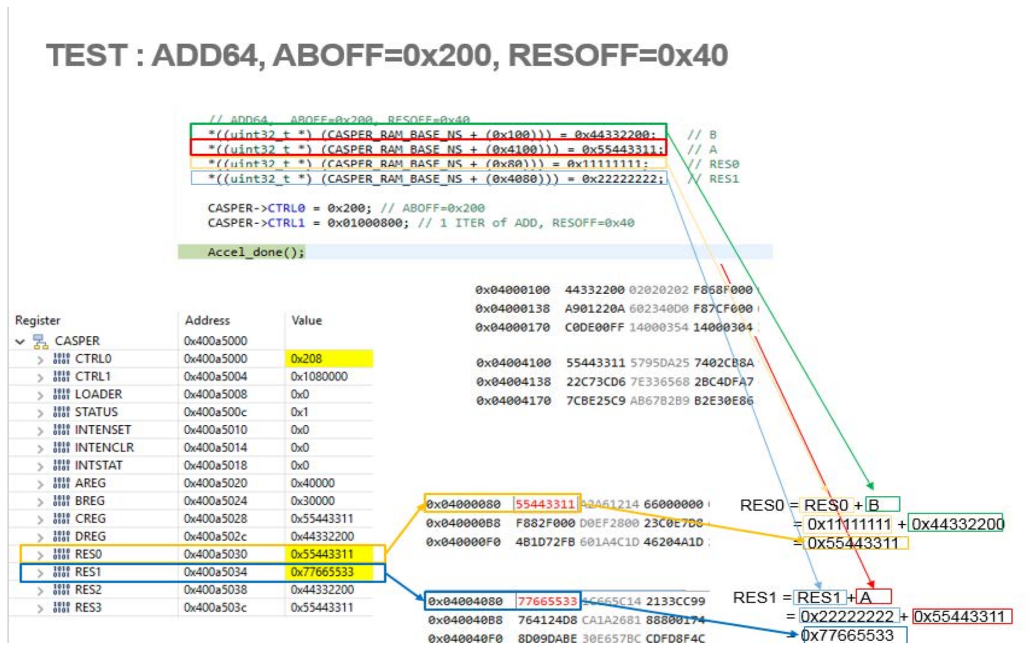


Fig 171. CASPER Memory map

This following example shows how CASPER registers are stored and used in memory:



TEST : ADD64, ABOFF=0x200, RESOFF=0x40

```
// ADD64,  ABOFF=0x200, RESOFF=0x40
*((uint32_t *) (CASPER_RAM_BASE_NS + (0x100))) = 0x44332200;    // B
*((uint32_t *) (CASPER_RAM_BASE_NS + (0x4100))) = 0x55443311;   // A
*((uint32_t *) (CASPER_RAM_BASE_NS + (0x80))) = 0x11111111;     // RES0
*((uint32_t *) (CASPER_RAM_BASE_NS + (0x4080))) = 0x22222222;   // RES1

CASPER->CTRL0 = 0x200; // ABOFF=0x200
CASPER->CTRL1 = 0x01000800; // 1 ITER of ADD, RESOFF=0x40

Accel_done();
```

Register	Address	Value
CASPER	0x400a5000	
CTRL0	0x400a5000	0x208
CTRL1	0x400a5004	0x1080000
LOADER	0x400a5008	0x0
STATUS	0x400a500c	0x1
INTENSET	0x400a5010	0x0
INTENCLR	0x400a5014	0x0
INTSTAT	0x400a5018	0x0
AREG	0x400a5020	0x40000
BREG	0x400a5024	0x30000
CREG	0x400a5028	0x55443311
DREG	0x400a502c	0x44332200
RES0	0x400a5030	0x55443311
RES1	0x400a5034	0x77665533
RES2	0x400a5038	0x44332200
RES3	0x400a503c	0x55443311

```
0x04000100   44332200 02020202 F868F000
0x04000138   A901220A 602340D0 F87CF000
0x04000170   C0DE00FF 14000354 14000304

0x04004100   55443311 5795DA25 7402CB8A
0x04004138   22C73CD6 7E336568 2BC4DFA7
0x04004170   7CBE25C9 AB67B2B9 B2E30E86

0x04000080   55443311 A2A61214 66000000
0x040000B8   F882F000 D0EF2800 23C0E7D8
0x040000F0   4B1D72FB 601A4C1D 46204A1D

0x04004080   77665533 1C665C14 2133CC99
0x040040B8   764124D8 CA1A2681 88800174
0x040040F0   8D09DABE 30E657BC CDFD8F4C
```

```
RES0 = RES0 + B
     = 0x11111111 + 0x44332200
     = 0x55443311

RES1 = RES1 + A
     = 0x22222222 + 0x55443311
     = 0x77665533
```

Fig 172. 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

45.3.1 CASPER co-processor operation

The ARMv8-M architecture (ARM M33) provides a co-processor interface that allows access to CASPER via the 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.

45.3.2 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.

45.3.3 CASPER modes

Coarsely integrated operand scanning = CIOS

Most Significant Word = MSW

Least Significant Word = LSW

Table 977. 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).
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.

Table 977. Casper AHB operations *…continued*

Mode	Name	Description	Comments
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.	

45.4 Register descriptions

Register descriptions for CASPER are provided in the following table.

Table 978. 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	45.4.1
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	45.4.2
LOADER	R/W	0x8	Contains an optional loader to load into CTRL0/1 in steps to perform a set of operations.	undefined	45.4.3
STATUS	R/W	0xC	Indicates operational status and would contain the carry bit if used.	undefined	45.4.4
INTENSET	R/W	0x10	Sets interrupts.	undefined	45.4.5
INTENCLR	R/W	0x14	Clears interrupts.	undefined	45.4.6
INTSTAT	R/W	0x18	Interrupt status bits (mask of INTENSET and STATUS).	undefined	45.4.7
AREG	R/W	0x20	A register.	undefined	45.4.8
BREG	R/W	0x24	B register.	undefined	45.4.8
CREG	R/W	0x28	C register.	undefined	45.4.8
DREG	R/W	0x2C	D register.	undefined	45.4.8
RES0	R/W	0x30	Result register 0.	undefined	45.4.9
RES1	R/W	0x34	Result register 1.	undefined	45.4.9
RES2	R/W	0x38	Result register 2.	undefined	45.4.9
RES3	R/W	0x3C	Result register 3.	undefined	45.4.9
MASK	R/W	0x60	Optional mask register.	undefined	45.4.10
REMASK	R/W	0x64	Optional re-mask register.	undefined	45.4.11
LOCK	R/W	0x80	Security lock register.	undefined	45.4.12

45.4.1 Control 0 pin register

Contains the offsets of AB and CD in the RAM.

Table 979. 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

45.4.2 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 980. 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 - 1. write 0 means does one cycle, does not iterate.	0x0
15:8	MODE		Iteration counter. Is number_cycles - 1. write 0 means does one cycle, 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).	
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.	

Table 980. 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
29	-	-	Reserved.	undefined
31:30	CSKIP		Skip rules on Carry if needed. This operation will be skipped based on Carry value (from previous operation) if not 0.	
		0	No Skip.	
		1	Skip if Carry is 1.	
		2	Skip if Carry is 0.	
		3	Set CTRLOFF to CDOFF and Skip.	

45.4.3 Loader register

Provides an optional loader to load into CTRL0/1 in order to perform a series of operations in succession.

Table 981. 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 Op 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.

45.4.4 Status register

Indicates operational status and, if used, contains the carry bit.

Table 982. 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 982. 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.

45.4.5 Interrupt set register

Sets the interrupts.

Table 983. 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.	

45.4.6 Interrupt clear register

Clears the interrupts.

Table 984. 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.	

45.4.7 Interrupt status bits register

Provides status for interrupts.

Table 985. 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.	

45.4.8 Data (A-D) registers

Specifies register to be fed to multiplier.

Table 986. 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

45.4.9 Result (0-3) registers

Holds working results for operation.

Table 987. 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

45.4.10 Mask register

Optional mask register used to apply a side channel countermeasure.

Table 988. 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

45.4.11 Re-mask register

Optional mask register used to apply a side channel countermeasure.

Table 989. 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

45.4.12 Security lock register

Reads back with security level locked to, or 0. Writes as 0 to unlock, 1 to lock.

Table 990. 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
		0	Unlock.	
		1	Lock to current security level.	
3:1	-		Reserved.	0x0
16:4	KEY		Must be written as 0x73D to change the register. 0011100111101b - If set during write, will allow lock or unlock.	0x0
31:17	-		Reserved.	0x0

UM11424

Chapter 46: LPC55S0x/LPC550x Debug Subsystem

Rev. 1.5 — 21 December 2023 **User manual**

46.1 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(TM) 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.

46.2 Features

- Supports arm serial wire debug mode for the CPU0.
- Trace port provides Cortex-M33 CPU instruction trace capability. Output is via a serial wire viewer.
- Direct debug access to all memories, registers, and peripherals.
- No target resources are required for the debugging session.
- Breakpoints: Includes eight instruction breakpoints.
- Watch-points: Includes four data watch-points that can also be used as triggers.
- Instrumentation Trace Macrocell allows additional software controlled trace for the CPU.

46.3 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 selecting the SWO function on either PIO0_8 or PIO0_10 and enabling the trace clock (see Section 46.4).

Debug access by a remote host is controlled by the LPC55s0x/LPC550x ROM and is only enabled when permitted through the device configuration and when the correct protocol is followed to initiate a debug session. For LPC550x devices, if the device has been configured for debug authentication, then a debug session must be initiated following the correct authentication sequence. When a device is in the development life-cycle state, the "Debug session protocol" described in Section 46.6 should be used. When the device is in deployed life-cycle state, the "Debug Authentication protocol" described in Section 46.7 should be used.

46.4 Pin description

Table 991 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 991. 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 4.5.44 “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 4.5.19 “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.

46.5 Debug Subsystem functional description

This section describes the hardware elements of the Debug Subsystem and defines the command protocols used by them. See Section 46.7 and Section 46.6 respectively for descriptions of how the host debug system interacts with the Debug Subsystem during debug sessions with and without debug authentication.

46.5.1 Debug subsystem components

Figure 173 shows the top-level debug ports and connections in the LPC55S0x/LPC550x. 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.
- 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.

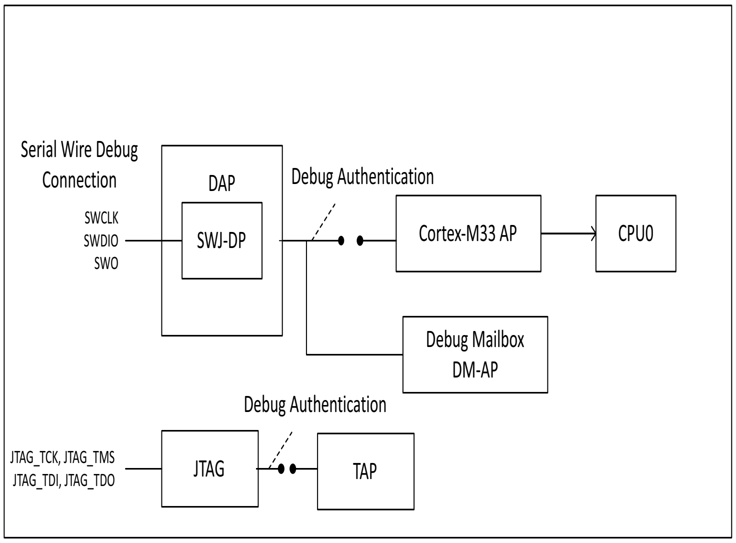


Fig 173. Internal debug system elements and connections

46.5.2 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 991. 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).

46.5.3 CPU0 AP

The DAP 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 46.7 “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 based on a 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.

46.5.4 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 46.7 “Debug authentication” for a description of the protocol used to initiate a debug session from a host debug system.

46.5.4.1 Register description

The registers in the debug mailbox are shown in Table 992. 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 992. Register overview: DBGMailbox (base address = 0x5009 C000)

Name	Access	Offset	Description	Reset value	Section
CSW	R/W	0x000	Command and status word.	0x0	46.5.4.1.1
REQUEST	R/W	0x004	Request from the debugger to the device.	0x0	46.5.4.1.2
RETURN	R/W	0x008	Return value from the device to the debugger.	0x0	46.5.4.1.3
ID	RO	0x0FC	Identification register.	0x002A 0000	46.5.4.1.4

46.5.4.1.1 Command and Status Word register

The CSW register contains command and status bits to facilitate communication between the debugger and the device.

Table 993. 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.	-

46.5.4.1.2 Request value register

The REQUEST register is used by a debugger to send action requests to the device.

Table 994. 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

46.5.4.1.3 Return value register

The RETURN register provides any response from the device to the debugger.

Table 995. 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

46.5.4.1.4 Identification register

The ID register provides an identification of the DM-AP interface.

Table 996. Identification register (ID, offset = 0x0FC)

Bit	Symbol	Description	Reset value
31:0	ID	Identification value.	0x002A0000

46.5.5 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. See Section 4.6.1 “Reset” for details on valid resets.

46.5.6 Mailbox commands

This section describes the Response and Request message formats and available mailbox commands and their associated parameters.

46.5.6.1 Request packet layout

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 997. 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[];
};
```

46.5.6.1.1 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” commands, 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 998. 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.
Bulk Erase (legacy command)	0x02	None	32-bit status	Erase the entire on-chip flash memory (excluding Protected Flash Regions).
Reserved	0x03	None	32-bit status	Reserved, Returns 3.
Exit DM_AP (legacy command)	0x04	None	32-bit status	Causes the device to continue normal debug flow.
Enter ISP Mode	0x05	dataWordCount: 0x1 data[0]: ISP mode enum. 0xffffffff - Auto detection 0x1 - UART 0x2 - I2C 0x4 - SPI 0x10 - Reserved. 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 situations 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 46.7 “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 46.7 “Debug authentication”.

Table 998. DM-AP commands *…continued*

Name	Command code	Parameters	Response	Description
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, 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	Starts 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.

46.5.6.2 Response packet layout

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 999. 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 999. 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[];
};
```

46.5.6.2.1 Response codes for DM-AP commands

The response codes for DM-AP commands are listed below. These codes can be read from the RETURN register (see Table 1000). 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 1000. 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.

46.5.6.3 ACK_TOKEN

The ACK_TOKEN provides an acknowledgement and is used in the following ways:

- 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 1001. 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 */
};
```

46.5.6.4 Set Fault Analysis (FA) mode command

LPC550x ROM offers the FA Mode (SET_FA_MODE) command handler to enable deletion of sensitive information (for example, Keys) before handing over the device to NXP for fault analysis. The ROM allows the SET_FA_MODE command only when corresponding flag in 'debug_state' is set.

Activation of the FA_MODE boot sequence will perform the following:

- Create a new version of *Customer Field Programmable* (CFPA) page.
- Set ENABLE_FA_MODE word in the page to value 0xC33CA55A.
- Erase all KEYS and IVs in KEYSTORE Flash page.
- Flush all temporary key registers.
- Blocks PUF indexes.
- Open all debug ports.
- Enter a while (1) loop.

46.5.6.5 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 needs to start with a new re-synchronization request in order to clear the error flag.

46.6 Debug session protocol

LPC55S0x/LPC550x Boot ROM implements a 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 (LPC550x only).

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 46.7 for full details regarding debug authentication.

On PC55s0x/LPC550x 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 to as 'Boot-ROM protection' in this manual. 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 sections 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 proceeds into ISP mode and waits 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”).

46.6.1 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 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).

The DM-AP commands are used to start a debug session and control the exchange of debug information. 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 998). 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. Result codes are shown in Table 1000. To handle situations where debugging 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 is loaded into RAM and used to program the application to be debugged, and sets the required breakpoints in the application code. 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.

46.6.2 Debug session with valid application in flash

When a debug session is associated with a valid application in flash, and has not been disabled by an application already in flash, 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 46.6.1 may be used in order to simplify debug support implementations.

46.6.3 Debug session attaching to a running target

In another scenario, a device has booted and is running an application that has not disabled the debug session, 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.

46.6.4 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 LPC55s0x/LPC550x 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.

The debugger uses the following reset sequence on these devices:

1. Set the data watch-point to halt the core on read of address location 0x50000040.
2. If all data watch points comparators are occupied, back-up one of the watch-point settings and replace it with the above watch-point location.
3. Reset the core and peripherals by setting the SYSRESETREQ bit in the AIRCR.
4. Wait for 100 msec to allow ROM to re-enable debug access.
5. Check the DHCSR register to see the core halted due to data watch-point.
6. If read of DHCSR times out or returns error response, then it indicates that the ROM entered ISP command handling loop due to invalid image in boot media.
   – Execute start debug session sequence described in Section 46.6.1 “Debug session with uninitialized/invalid flash image or ISP mode”.
   – Wait for 10 msec.
   – Now the CM33_AP should be enabled.
   – Read DHCSR and issue HALT to CM33_AP.
7. Clear the data watch-point added in step 1. If a watch-point was set as described in step 2, then restore the watch-point configuration.

46.7 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 LPC550x 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 LPC550x 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 174 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.

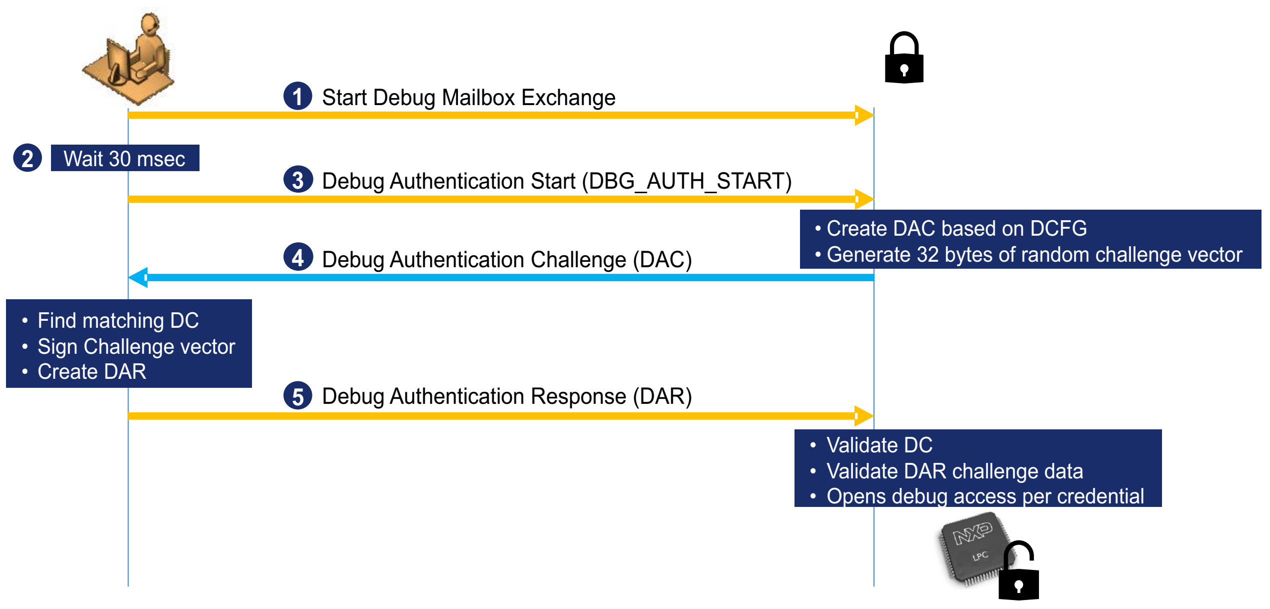

1 Start Debug Mailbox Exchange
2 Wait 30 msec
3 Debug Authentication Start (DBG_AUTH_START)
• Create DAC based on DCFG
• Generate 32 bytes of random challenge vector
4 Debug Authentication Challenge (DAC)
• Find matching DC
• Sign Challenge vector
• Create DAR
5 Debug Authentication Response (DAR)
• Validate DC
• Validate DAR challenge data
• Opens debug access per credential
NXP
LPC

Fig 174. Debug authentication flow

46.7.1 Debug Access Control Configuration

The boot code present inLPC550x 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.

46.7.1.1 Protocol Version (DCFG_VER)

The LPC550x 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 (0x3E41C) 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.

46.7.1.2 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 “LPC55S0x 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 0x3E450 – 0x3E46C specifies the ROTID.

- CFPA word ROTKH_REVOKE (at offset 0x18).

46.7.1.3 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 0x3E410) 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

46.7.1.4 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 1003 "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 1002.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 46.7.6.2 "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 LPC551x (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 LPC551x are shown in Table 1003.

Table 1003.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	-	Reserved.
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	-	Reserved.
31:9	-	Reserved

Table 1004.Layout of CC_SOCU_PIN (0x3E410) & CC_SOCU_PIN_NS (CFPA offset 0x020)

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 1003 “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 1005.Layout of CC_SOCU_DFLT (0x3E414) & CC_SOCU_DFLT_NS (CFPA offset 0x024)

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 1003 “CC_LIST_Table”.
31:16	INV_SOCU_DFLT[n]	Inverse value of the above bitfield [15:0].

46.7.1.5 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 LPC551x provides 4 bytes length DCFG_VENDOR_USAGE field, composed from following fields:

- Upper 2 bytes from CMPA.VENDOR_USAGE (PFR address 0x9E418 for LPC551x use 0x3E418)

  - 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.

46.7.2 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).

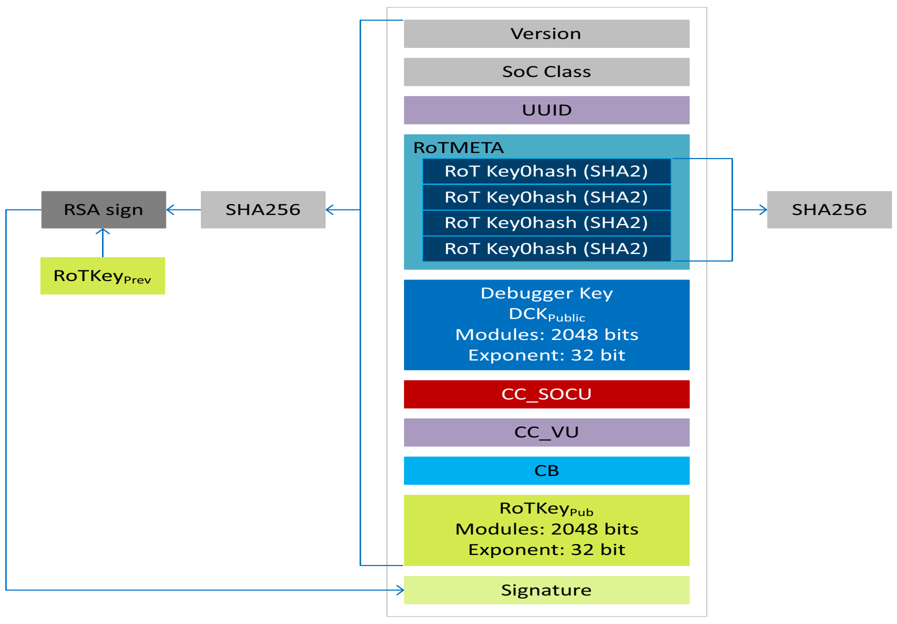


Fig 175. Debug Credential certificate fields

The data structure is a represented as a packed binary concatenation of its component fields as shown in the list below:

Table 1006.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 46.7.1.5 “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

46.7.3 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 176.

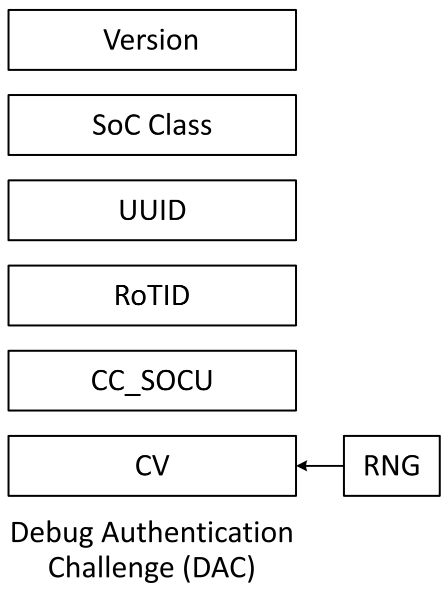


Fig 176. Debug Authentication Challenge (DAC) fields

Table 1007. 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

Table 1007.Debug Authentication Challenge (DAC) fields *…continued*

Name	Offset	Description	Size in bytes
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 46.7.1.2 “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 1003 “CC_LIST_Table” for bit encoding of this field. • A 32-bit SOCU_DFLT value. See Table 1003 “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

46.7.4 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.

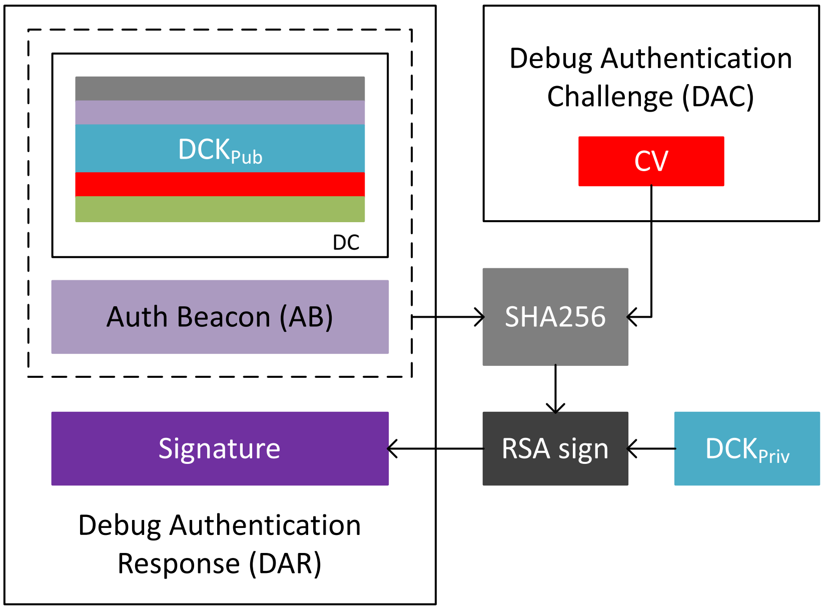

DCK$_{Pub}$
DC
Auth Beacon (AB)
Signature
Debug Authentication Response (DAR)
Debug Authentication Challenge (DAC)
CV
SHA256
RSA sign
DCK$_{Priv}$

Fig 177. Debug Authentication Response (DAR) fields

Table 1008.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 46.7.2 "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 46.7.2 "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

46.7.5 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.

46.7.5.1 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.

46.7.6 41.9.6Debug Authentication Use cases

46.7.6.1 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.

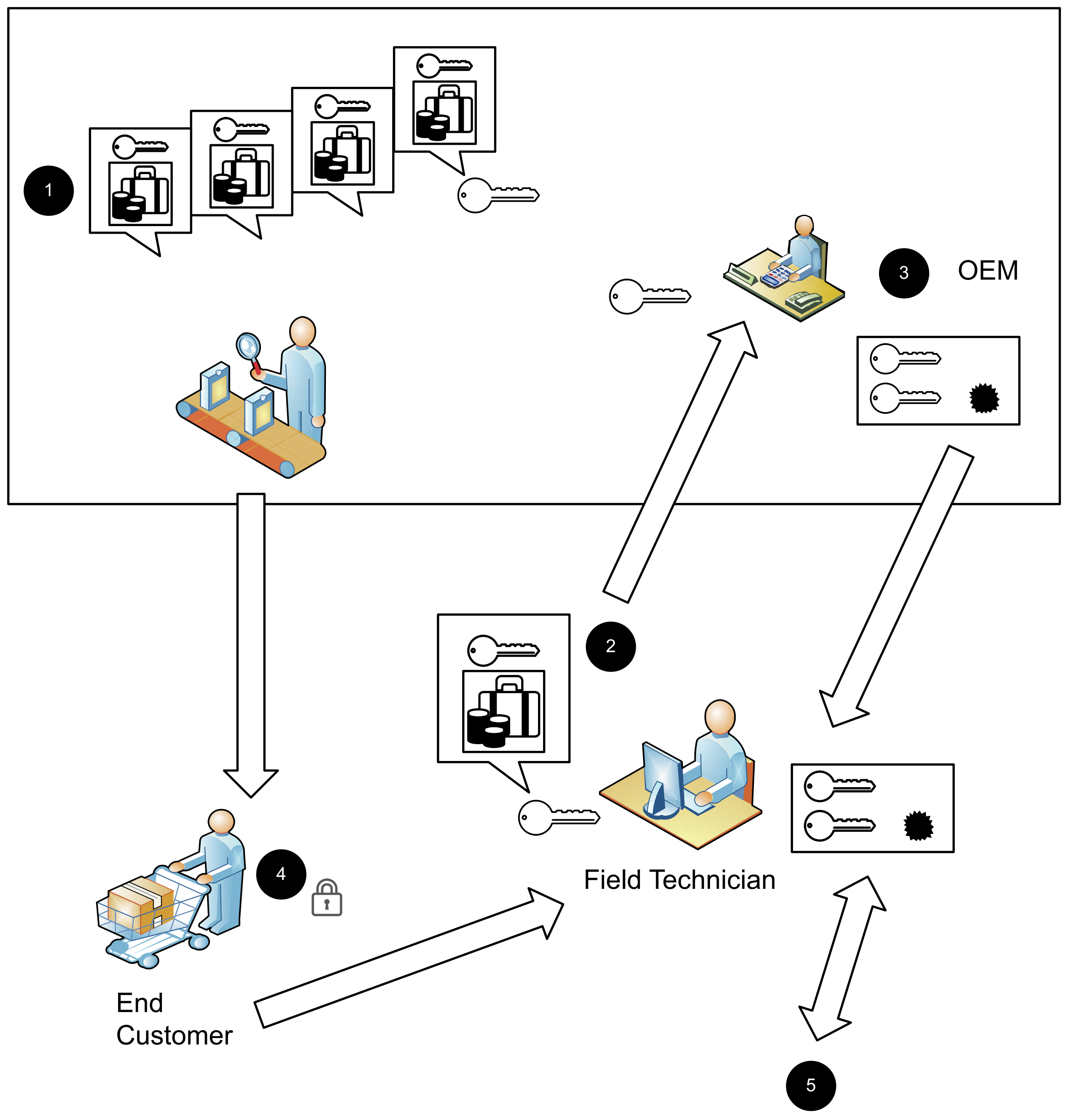


Fig 178. Debug Authentication protocol usage example

46.7.6.2 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.

46.7.7 41.9.7Glossary

Table 1009.41.9.7Glossary

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 1009.41.9.7Glossary *…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.
CPFA		Customer Programmable Factory area.
CPIA		Customer Programmable in-field area.

UM11424

Chapter 47: Supplementary information

Rev. 1.5 — 21 December 2023 User manual

47.1 Abbreviations

Table 1010.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.
CAN FD	Controller Area Network Flexible Data Rate.
CM33	Cortex M33.
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.
SPI	Serial Peripheral Interface.

Table 1010.Abbreviations *…continued*

Acronym	Description
SRAM	Static Random Access Memory.
SWD	Serial-Wire Debug.
TAP	Test Access Port.
USART	Universal Synchronous/Asynchronous Receiver/Transmitter.
VAD	Voice Activity Detect.

47.2 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

47.3 Tables

47.4 Figures

47.5 Contents

Chapter 9: LPC55S0x/LPC550x Flash API

Chapter 10: LPC55S0x/LPC550x Protected Flash Region

Chapter 11: LPC55S0x/LPC550x Analog control

Chapter 12: LPC55S0x/LPC550x Cap Bank API

Chapter 13: LPC55S0x/LPC550x Power Management

Chapter 19: LPC55S0x/LPC550x Pin Interrupt and Pattern Match (PINT)

Chapter 20: LPC55S0x/LPC550x Secure pin interrupt and pattern match (Secure PINT)

Chapter 21: LPC55S0x/LPC550x:Group GPIO Input Interrupt (GINT0/1)

Chapter 22: LPC55S0x/LPC550x DMA controller

Chapter 24: LPC55S0x/LPC550x Standard counter/timers (CTIMER0 - 4)

Chapter 25: LPC55S0x/LPC550x Micro-tick Timer (UTICK)

Chapter 26: LPC55S0x/LPC550x Multi-Rate Timer

Chapter 27: LPC55S0x/LPC550x Real-Time Clock (RTC)

Chapter 28: LPC55S0x/LPC550x System Tick Timer

Chapter 29: LPC55S0x/LPC550x Windowed Watchdog Timer (WWDT)

Chapter 30: LPC55S0x/LPC550x Code Watchdog Timer

Chapter 31: LPC55S0x/LPC550x OS Event Timer

Chapter 32: LPC55S0x/LPC550x Flexcomm Interface Serial Communication

Chapter 42: LPC55S0x/LPC550x CRC engine

Chapter 44: LPC55S0x Security features

Chapter 47: 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.

Trademarks

Date of release: 12/21/2023
Document identifier: UM11424