

MCXW23RM

MCX W23 for IoT applications

Rev. 3.0 — 17 October 2025

Reference manual

Document information

Information	Content
Keywords	SmartSensor, IoT, Healthcare, Bluetooth Low Energy
Abstract	This user manual describes the MCX W23 for IoT applications.



1 Introduction

The MCX W23 is a family of devices. All devices are Arm Cortex®-M33 based wireless microcontrollers for embedded applications supporting Bluetooth Low Energy 5.3. It can be used to develop IoT solutions.

MCX W23xA supports LV_SM mode. MCX W23xB supports HV_SM and XR_SM mode.

These devices include:

- Up to 128 kB of on-chip SRAM
- Up to 1024 kB on-chip flash
- Quad SPI interface for operation from external SPI NVM
- Five general-purpose timers (CTIMER)
- One SCTimer/PWM
- One RTC/alarm timer
- One 24-bit multirate timer (MRT)
- Windowed watchdog timer (WWDT)
- Three flexible serial communication peripherals (each of which can be a USART, SPI, or I²C interface)

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 MCX W23 also offers support for SHA-1, SHA2-256, AES, RSA, ECC, UUID, dynamic encryption, and decryption of the flash data using a PRINCE engine, debug authentication, and TBSA-M compliance.

1.1 Features

- Arm Cortex-M33 core (r0p4):
 - Running at a frequency of up to 32 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 (SWD) with eight breakpoints and four watch points. Includes serial wire output (SWO) for enhanced debug capabilities.
 - System tick timer.
- On-chip memory:
 - MCX W235x: 512 kB , MCX W236x: 1 MB of flash with cache and error correction code management, including single bit correction and double bit fault detection
 - MCX W235x: 96 kB, MCX W236x: 128 kB of RAM
 - Dedicated packet RAM buffer for radio subsystem: 16 kB data RAM and 512 bytes sequencer RAM
 - 256 kB of ROM
- On-chip ROM bootloader supports:
 - Booting of images from on-chip flash
 - Image integrity checking (signature and CRC32)
 - Flash programming through in-system programming (ISP) commands over the following interfaces:
 - SPI slave interfaces (Flexcomm 0) using mode 3 (CPOL = 1 and CPHA = 1)
 - I2C slave interface (Flexcomm 1)
 - UART interface (Flexcomm 2) with autobaud
 - ROM API functions:
 - Flash programming API
 - MBED TLS API

- Secure firmware update API using NXP secure boot file format, version 2.1 (SB2 files).
 - Preloading a TrustZone configuration before the application boots
 - Booting of images from PRINCE encrypted flash regions
 - NXP debug authentication protocol (DAP), versions 1.0 (RSA-2048) and 1.1 (RSA-4096)
 - Setting a sealed part to fault analysis mode through debug authentication
 - Boot seed as described by Arm platform security architecture specification
 - OEM key provisioning during device manufacturing
- 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 X.509 certificate format to validate image public keys
 - Supports up to four revocable root of trust (RoT) or certificate authority (CA) keys
The root of trust (RoT) establishment is achieved by storing the SHA-256 hash digest of the SHA-256 hashes of (at most) four RoT public keys in protected flash region (PFR).
 - Supports up to 16 image key certificate revocations using serial number field in X.509 certificate
 - Supports antirollback feature for both RoT and image key revocation
- The MCX W23 is compatible with the Bluetooth Low Energy 5.3 specification:
 - Bluetooth Low Energy 5.3 controller subsystem (QDID 200592)
 - Bluetooth Low Energy 5.3 host subsystem (QDID 226395)
 - Includes a 48-bit unique bluetooth device address
 - Up to 4 simultaneous connections supported
- The MCX W23 supports in the following Bluetooth Low Energy features:
 - Device privacy and network privacy modes (version 5.0)
 - Advertising extension PDUs (version 5.0)
 - Anonymous device address type (version 5.0)
 - Up to 2 Mbps data rate (version 5.0)
 - Long range (version 5.0)
 - High-duty cycle, non-connectable advertising (version 5.0)
 - Channel selection algorithm #2 (version 5.0)
 - High output power (version 5.0)
 - Advertising channel index (ACI) (version 5.1)
 - Periodic advertising sync transfer (PAST) (version 5.1)
 - Low energy (LE) power control feature (version 5.2)
- Serial interfaces:
 - The Flexcomm interface contains up to three serial peripherals (Flexcomm interface 0-2). Each Flexcomm interface can be a USART, an SPI, and a I2C interface (selected by software). Each Flexcomm interface includes a FIFO that supports USART and SPI. Various clocking options are available to each Flexcomm interface, including a shared fractional baud rate generator and a timeout feature.
 - The I2C bus interfaces support fast mode and fast mode plus with data rates of up to 1 Mbit/s and multiple address recognition and monitor modes.
 - Quad SPI interface for external serial flash support with execute-in-place (XIP).
- Digital peripherals:
 - DMA0 controller with 8 channels and up to 21 programmable triggers, able to access all memories and DMA-capable peripherals
 - DMA1 controller with 8 channels and up to 15 programmable triggers, able to access all memories and DMA-capable peripherals
 - Up to 23 general-purpose input/output (GPIO) pins

- GPIO registers are 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.
- One 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:
 - Armv8-M TrustZone-M® enabled
 - AES-256 encryption/decryption engine, using keys fed directly from PUF or using a software-supplied key
 - Secure hash algorithm (SHA1 and SHA2-256) module supports secure boot with dedicated DMA controller
 - Physical unclonable function (PUF) using dedicated SRAM for silicon fingerprint; PUF can generate, store, and reconstruct keys with sizes ranging from 64 bits to 4096 bits. Includes hardware for key extraction.
 - True random number generator (TRNG) compliant to NIST SP 800-90B, NIST SP 800-22, and BSI AIS-31
 - 128-bit unique device serial number for identification (universal unique identifier: UUID)
 - Secure GPIO
 - Public key coprocessor (PKC) is provided to enable hardware acceleration for various functions required for certain asymmetric cryptographic algorithms, such as, elliptic curve cryptography (ECC) or RSA.
 - PRINCE module for real-time encryption of data written to on-chip flash and real-time decryption of encrypted flash data during read to allow asset protection, such as securing application code and to enable secure flash update
 - Protected flash region (PFR) available to configure secure boot, debug authentication, read UUID, store PUF in key store area, and user-defined fields available for specific data storage
- 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 1 s resolution running in the always-on power domain. Another timer in the RTC can be used for waking up from all low-power modes including deep-power down, with a 1 ms resolution. The RTC is clocked by the 32.768 kHz FRO or 32.768 kHz external crystal.
 - Multiple-channel multirate 24-bit timer (MRT) for repetitive interrupt generation at up to four programmable, fixed rates
 - Windowed watchdog timer (WWDT) with FRO 1 MHz as clock source.
 - The micro-tick timer running from the watchdog oscillator can be used to wake up the device from sleep and deep-sleep modes. Includes 4 capture registers with pin inputs.
 - 42-bit free running OS timer as continuous timebase for the system, available in any reduced power modes (except POWEROFF state). It runs on a 32 kHz clock source, allowing a count period of more than four years.
- Clock generation:
 - Internal free-running oscillator (FRO)
This oscillator provides a selectable 32 MHz output and a 12 MHz output (divided down from the selected higher frequency) that can be used as a system clock. The FRO is trimmed to ± 2 % accuracy over the entire voltage and temperature range.
 - 32.768 kHz FRO. The FRO is trimmed to ± 2 % accuracy over the entire voltage and temperature range
 - Internal low-power oscillator (FRO 1 MHz) trimmed to ± 15 % accuracy over the entire voltage and temperature range
 - Crystal oscillator with an operating frequency of 32 MHz. (± 40 ppm)

- Crystal oscillator with 32.768 kHz operating frequency
- 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.
- Flexible power management
 - MXCW23xA
 - Low-voltage supply mode (LV_SM) for 1.5 V battery support using integrated boost DC-DC converter
 - MXCW23xB
 - High-voltage supply mode (HV_SM) for 3 V battery support using integrated buck DC-DC converter
 - External regulated supply mode (XR_SM) for PMIC support (bypassing the integrated DC-DC converter)
- Power-saving and wake-up:
 - Integrated power management controller (PMC) to minimize power consumption
 - Reduced power modes:
 - Sleep, deep-sleep with RAM retention, power-down with RAM retention and CPU retention, deep power-down with RAM retention and power-off.
 - 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 any low-power mode (except POWEROFF state).
 - 2 brownout detectors (BOD) for VBAT_HV or VBAT_LV voltage with separate thresholds for forced reset or interrupt generation
- JTAG boundary scan supported
- Operating temperature range: –40 °C to +85 °C
- Available in HVQFN40 and WLCSP37 packages

1.2 Block diagram

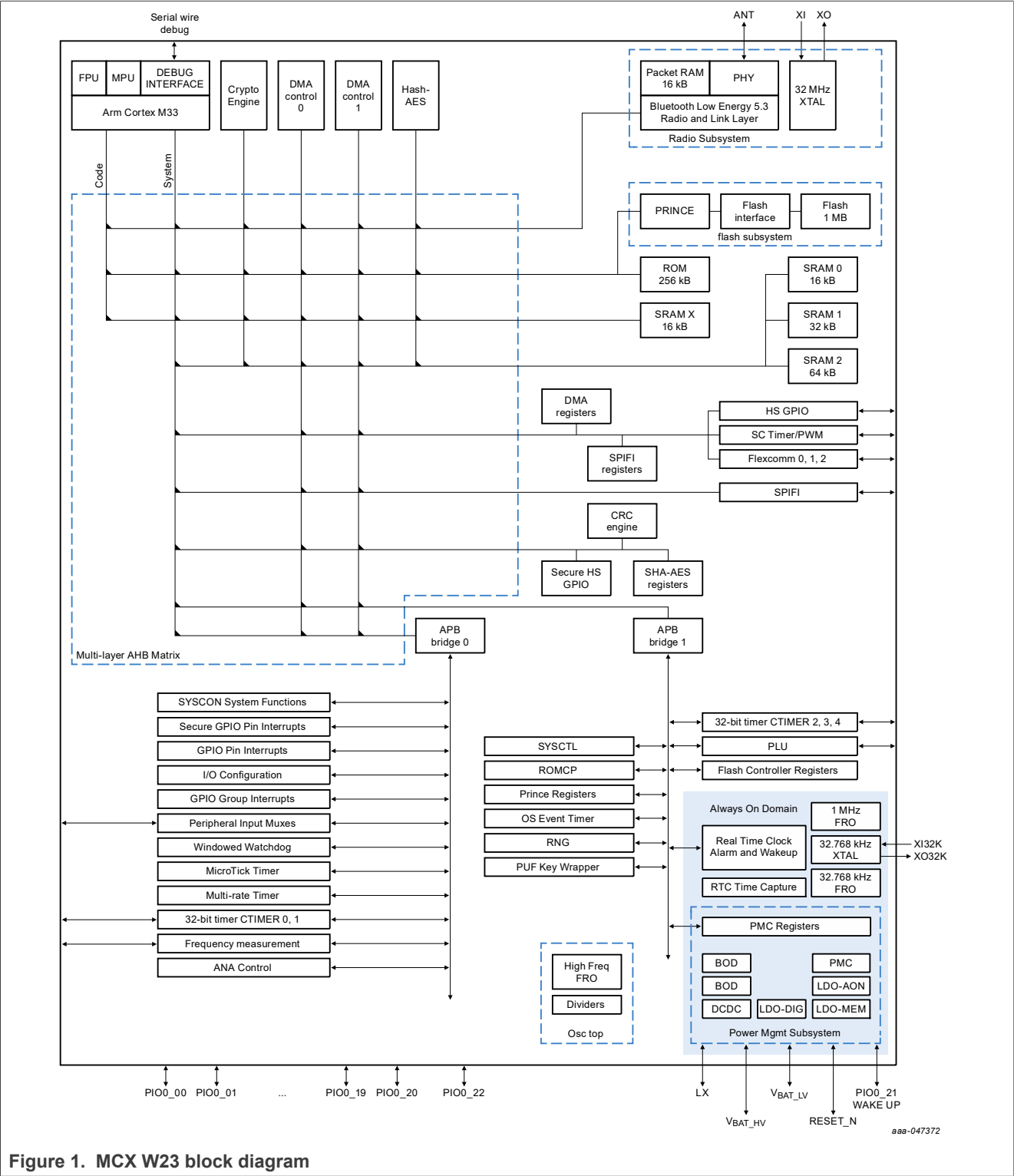


Figure 1. MCX W23 block diagram

1.3 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 MCX W23 uses a multilayer AHB matrix. It connects the Arm Cortex M33 buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing that different bus masters access peripherals that are on different slave ports of the matrix simultaneously.

1.4 Arm Cortex-M33 with 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 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
- Multiple core buses that can be accessed simultaneously

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

1.5 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.

1.6 On-chip static RAM

The MCX W23 supports various configurations of SRAM with separate bus master accesses for higher throughput and individual power control for low-power operation.

1.7 Ordering information

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
MCXW236BIHNAR	HVQFN40	plastic low profile quad flat package; 40 leads; body 5 × 5 × 0.85, 0.4 mm pitch	SOT1369-5
MCXW236BIUKAR	WLCSP37	wafer level chip scale package; 37 bumps; body 2.63 × 2.495 × 0.49, 0.35 mm pitch	SOT2173-1
MCXW236AIHNAR	HVQFN40	plastic low profile quad flat package; 40 leads; body 5 × 5 × 0.85, 0.4 mm pitch	SOT1369-5
MCXW236AIUKAR	WLCSP37	wafer level chip scale package; 37 bumps; body 2.63 × 2.495 × 0.49; 0.35 mm pitch	SOT2173-1
MCXW235BIHNAR	HVQFN40	plastic low profile quad flat package; 40 leads; body 5 × 5 × 0.85, 0.4 mm pitch	SOT1369-5
MCXW235BIUKAR	WLCSP37	wafer level chip scale packages; 37 bumps; body 2.63 × 2.495 × 0.49; 0.35 mm pitch	SOT2173-1
MCXW235AIUKAR	WLCSP37	wafer level chip scale packages; 37 bumps; body 2.63 × 2.495 × 0.49; 0.35 mm pitch	SOT2173-1
MCXW235AIHNAR	HVQFN40	plastic low profile quad flat package; 40 leads; body 5 × 5 × 0.85, 0.4 mm pitch	SOT1369-5

1.7.1 Ordering options

Table 2. Ordering options

Type number	Supply mode	Maximum CPU frequency (MHz)	Public key coprocessor	Flash/KB	Total SRAM/KB	Secure boot	TrustZone	PRINCE	PUF controller	HASH-AES	GPIO
MCXW236BIHNAR	HV_SM XR_SM	32	yes	1024	134.5 (128 + 16.5)	yes	yes	yes	yes	yes	23
MCXW236BIUKAR	HV_SM XR_SM	32	yes	1024	134.5 (128 + 16.5)	yes	yes	yes	yes	yes	23
MCXW236AIHNAR	LV_SM	32	yes	1024	134.5 (128 + 16.5)	yes	yes	yes	yes	yes	23
MCXW236AIUKAR	LV_SM	32	yes	1024	134.5 (128 + 16.5)	yes	yes	yes	yes	yes	23
MCXW235BIHNAR	HV_SM XR_SM	32	yes	512	112.5(96 + 16.5)	yes	yes	yes	yes	yes	23
MCXW235BIUKAR	HV_SM XR_SM	32	yes	512	112.5(96 + 16.5)	yes	yes	yes	yes	yes	23
MCXW235AIHNAR	LV_SM	32	yes	512	112.5(96 + 16.5)	yes	yes	yes	yes	yes	23

Table 2. Ordering options...continued

Type number	Supply mode	Maximum CPU frequency (MHz)	Public key coprocessor	Flash/KB	Total SRAM/KB	Secure boot	TrustZone	PRINCE	PUF controller	HASH-AES	GPIO
MCXW235AIUKAR	LV_SM	32	yes	512	112.5(96 + 16.5)	yes	yes	yes	yes	yes	23

2 Memory map

2.1 AHB multilayer matrix

The MCX W23 uses a multilayer AHB matrix to connect the CPU buses and other bus masters to peripherals in a flexible manner. It optimizes performance by allowing different bus masters to access peripherals that are on different slave ports of the matrix simultaneously. [Figure 1](#) shows details of the available matrix connections.

2.2 Memory protection unit (MPU)

The CPU has a memory protection unit (MPU) for the secure and the non-secure world. It 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.

2.3 TrustZone and system mapping on this device

The IDAU of the trusted execution environment (TEE; external to the CPU) uses address bit 28 to divide the address space into potential secure and non-secure regions. Memory access hardware ignores address bit 28. So, each physical location appears in two places whatever bus they are located on. SAU of the Arm TrustZone (inside the CPU) determines which CPU accesses (including non-secure callable) are 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.

Note: 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 required.

Table 3. TrustZone and system general mapping^{[1][2]}

Start address	End address	TrustZone, CPU0 only	CPU bus	CM-33 usage
0x0000 0000	0x0FFF FFFF	non-secure	code	flash memory, boot ROM, SRAM X
0x1000 0000	0x1FFF FFFF	secure	code	
0x2000 0000	0x2FFF FFFF	non-secure	data	SRAM 0, SRAM 1, SRAM 2, SPIFI
0x3000 0000	0x3FFF FFFF	secure	data	
0x4000 0000	0x4FFF FFFF	non-secure	data	AHB and APB peripherals
0x5000 0000	0x5FFF FFFF	secure	data	

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

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

2.4 Memory map overview

[Table 4](#) gives a more detailed memory map, as the Cortex-M33 (CPU) sees it. The purpose of the four address spaces for the shared RAMs is outlined at the beginning of this chapter. The mapping of the shared RAM regions on the 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	0x000F FFFF	0x1000 0000	0x100F FFFF	flash memory on CM33 code bus, 1 MB (MCX W236x only)
	0x0000 0000	0x0007 FFFF	0x1000 0000	0x1007 FFFF	flash memory on CM33 code bus, 512 kB (MCX W235x only)
	0x0300 0000	0x0303 FFFF	0x1300 0000	0x1303 FFFF	boot ROM on CM33 code bus
1	0x0400 0000	0x0400 3FFF	0x1400 0000	0x1400 3FFF	SRAM X on CM33 code bus, 16 kB The public key coprocessor (PKC) uses SRAMX_0 (0x1400 0000 to 0x1400 0FFF) and SRAMX_1 (0x1400 1000 to 0x1400 1FFF) (total: 8 kB). If CPU retention is used during power-down mode, SRAMX_2 is used by default in the power API (requires 1244 bytes). The start address is user-configurable within SRAMX_2 (0x1400 2000 to 0x1400 2FFF) and SRAMX_3 (0x1400 3000 to 0x1400 3FFF).
2	0x2000 4000	0x2000 7FFF	0x3000 4000	0x3000 7FFF	SRAM 0 on CM33 data bus; 16 kB
3	0x2000 8000	0x2000 FFFF	0x3000 8000	0x3000 FFFF	SRAM 1 on CM33 data bus; 32 kB
4	0x2001 0000	0x2001 FFFF	0x3001 0000	0x3001 FFFF	SRAM 2 on CM33 data bus; 64 kB. (MCX W236x only)
	0x2001 0000	0x2001 7FFF	0x3001 0000	0x3001 7FFF	SRAM 2 on CM33 data bus; 32 kB. (MCX W235x only)
5	0x2800 0000	0x2FFF FFFF	0x3800 0000	0x3FFF FFFF	SPIFI slave port 1 (memory) aperture of 128 MB
6	0x4000 0000	0x4001 FFFF	0x5000 0000	0x5001 FFFF	AHB to APB bridge 0 (see Section 2.5).
	0x4002 0000	0x4003 FFFF	0x5002 0000	0x5003 FFFF	AHB to APB bridge 1 (see Section 2.5)
7	0x4008 0000	0x4008 FFFF	0x5008 0000	0x5008 FFFF	AHB peripherals (see Section 2.6)
8	0x4009 0000	0x4009 FFFF	0x5009 0000	0x5009 FFFF	AHB peripherals (see Section 2.6)
9	0x400A 0000	0x400A FFFF	0x500A 0000	0x500A FFFF	AHB peripherals (see Section 2.6)
10	0x400B 0000	0x400B FFFF	0x500B 0000	0x500B FFFF	AHB peripherals (see Section 2.6)

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

2.5 APB peripherals

[Table 5](#) provides details of the addresses for APB peripherals. APB peripherals have secure and non-secure access possibilities.

Table 5. APB peripherals memory map

APB bridge	Non-secure base address	Secure base address	Peripheral
0	0x4000 0000	0x5000 0000	system control (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	reserved
	0x4000 4000	0x5000 4000	pin interrupt and pattern match (PINT)
	0x4000 5000	0x5000 5000	secure pin interrupt and pattern match (SECPINT)
	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 (WWDTO)
	0x4000 D000	0x5000 D000	multirate timer (MRT)
	0x4000 E000	0x5000 E000	micro-tick timer (UTICK)
	0x4001 3000	0x5001 3000	analog controls (ANACTRL)
	0x4001 5000	0x5001 5000	reserved
1	0x4002 3000	0x5002 3000	auxiliary flash and random number generator control (SYSCTL)
	0x4002 8000	0x5002 8000	standard counter/timer 2 (CTIMER2)
	0x4002 9000	0x5002 9000	standard counter/timer 3 (CTIMER3)
	0x4002 A000	0x5002 A000	standard counter/timer 4 (CTIMER4)
	0x4002 C000	0x5002 C000	real-time clock and wake-up timer (RTC)
	0x4002 D000	0x5002 D000	operating system and event timer (OSTIMER)
	0x4003 4000	0x5003 4000	flash controller (MSF1)
	0x4003 5000	0x5003 5000	PRINCE dynamic encryption engine (NPX)
	0x4003 A000	0x5003 A000	true random number generator (TRNG)
	0x4003 B000	0x5003 B000	physical unclonable function (PUF)
	0x4003 D000	0x5003 D000	programmable logic unit (PLU)

2.6 AHB peripherals

[Table 6](#) provides details of the addresses for AHB peripherals. AHB peripherals have both secure and non-secure access possibilities.

Table 6. AHB peripheral memory map

AHB port	Non-secure base address	Secure base address	Peripheral
7	0x4008 2000	0x5008 2000	DMA controller 0 (DMA0)
	0x4008 5000	0x5008 5000	state configurable timer/PWM (SCT0)
	0x4008 6000	0x5008 6000	Flexcomm interface 0 (FLEXCOMM0)
	0x4008 7000	0x5008 7000	Flexcomm interface 1 (FLEXCOMM1)
	0x4008 8000	0x5008 8000	Flexcomm interface 2 (FLEXCOMM2)
	0x4008 B000	0x5008 B000	reserved
	0x4008 C000	0x5008 C000	general-purpose IO (GPIO)
8	0x4009 5000	0x5009 5000	cyclic redundancy check engine (CRC)
	0x4009 B000	0x5009 B000	reserved
	0x4009 C000	0x5009 C000	debug mailbox (DBGMAILBOX)
	0x4009 F000	0x5009 F000	SPIFI slave port 0 (SPIFI0)
9	0x400A 4000	0x500A 4000	hash-AES engine (HASHCRYPT)
	0x400A 5000	0x500A 5000	public key coprocessor (CASPER)
	0x400A 7000	0x500A 7000	DMA controller 1 (DMA1)
	0x400A 8000	0x500A 8000	secure GPIO (SECGPIO)
	0x400A C000	0x500A C000	security control registers (AHB_SECURE_CTRL)

2.7 SRAM

[Table 7](#) describes the RAM configuration for the MCX W23.

Table 7. SRAM configuration

Device type	RAM-X (KB)	RAM0 (KB)	RAM1 (KB)	RAM2 (KB)	Bluetooth Low Energy RAM (KB)
MCXW236x	16	16	32	64	16 + 0.5
MCXW235x	16	16	32	32	16 + 0.5

3 Nested vectored interrupt controller (NVIC)

Available interrupt sources may vary with specific MCX W23 device types.

3.1 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 of the Cortex-M33 supports:
 - 64 vectored interrupt slots
 - Eight programmable interrupt priority levels with hardware priority level masking
 - Vector table offset register VTOR
 - Software interrupt generation
 - Support for NMI from any interrupt, see [Section 21](#).

3.2 General description

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

3.2.1 Interrupt sources

[Table 8](#) lists the interrupt sources for each peripheral function. Each peripheral device may have one or more interrupt lines to the vectored interrupt controller. Each line may represent more than one interrupt source. The interrupt number does not imply any interrupt priority when interrupts are not given the same priority. In the case, where two interrupts, for example, are given the same priority, then the interrupt number below is relevant.

For detailed descriptions of the NVIC and the NVIC registers, see the Arm Cortex-M33 Devices Generic User Guide ([Ref. 1](#)).

Table 8. Connection of interrupt sources to the NVIC

Interrupt	Name	Interrupt description	Flags
0	WDT_BOD_IRQn	Device-specific interrupts. A combination of multiple interrupt sources: • code watchdog • windowed watchdog timer • brownout detect • DCDC status • flash controller • flash security violation • AHB security violation	-
1	DMA0_IRQn	DMA0 controller	interrupt A and interrupt B; error interrupt
2	GINT0_IRQn	GPIO group 0	enabled pin interrupts
3	reserved	-	-
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...continued

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	UTICK_STAT_INTR
9	MRT0_IRQn	multirate 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)	see enable read and set register of this module.
15	FLEXCOMM1_IRQn	Flexcomm interface 1 (USART, SPI, I ² C).	
16	FLEXCOMM2_IRQn	Flexcomm interface 2 (USART, SPI, I ² C)	
17	reserved	-	-
18	reserved	-	-
19	reserved	-	-
20	reserved	-	-
21	reserved	-	-
22	BLE_LL_IRQn	Bluetooth Low Energy link layer	Bluetooth Low Energy link-layer interrupt; can only be issued when bus clock is present
23	BLE_SLP_TMR_IRQn	Bluetooth Low Energy sleep timer	Bluetooth Low Energy sleep timer interrupt; can also be issued when bus clock is not present
24	WDT_IRQn	windowed watchdog timer	WDINT (watchdog warning interrupt flag) WDTOF (watchdog time-out flag)
25	reserved	-	-
26	reserved	-	-
27	BOD1_IRQn	BOD 1	-
28	BOD2_IRQn	BOD 2	-
29	RTC_IRQn	RTC alarm and wake-up interrupts	see enable read and set register of this module.
30	reserved	-	-
31	WAKE_DSLP_IRQn	WAKEUP interrupt	Wake-up interrupt when returning from DEEP SLEEP

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

Interrupt	Name	Interrupt description	Flags
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
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	SPIFI_IRQn	SPIFI interrupt	SPIFI_STAT_INTRQ
40	reserved	-	-
41	reserved	-	-
42	reserved	-	-
43	FLASH_CCIF_IRQn	FLASH command completed	FMU_FSTAT_CCIF - Flash command complete interrupt flag
44	FLASH_DFIF_IRQn	FLASH dual fault detect	FMU_FSTAT_DFDIF - Flash ECC unrecoverable double-bit error detect
45	reserved	-	-
46	reserved	-	-
47	reserved	-	-
48	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). Two pin interrupt outputs are available to NVIC.	SGPIO 0 interrupts
51	SEC_GPIO_INT0_IRQ1_IRQn	Secure GPIO function is available on P0(0-31). Two 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_IRQn	Cryptocoprocessor interrupt	Cryptocoprocessor interrupts
56	PUF_IRQn	PUF controller interrupt	PUF interrupts
57	TRNG_IRQn	random number generator	random number generator interrupts
58	DMA1_IRQn	DMA1 controller	DMA interrupts
59	reserved	-	-

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

Interrupt	Name	Interrupt description	Flags
60	CDOG_IRQn	code watchdog	CDOG_FLAGS

4 System Control (SYSCON)

4.1 Features

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

4.2 Basic configuration

Configure the SYSCON block as follows:

- No clock configuration is required. The clock to the SYSCON block is always enabled. By default, the FRO 12 MHz (fro_12m) clocks the SYSCON block.
- Target and reference clocks for the frequency measurement function are selected in the input mux block (see [Section 18](#)).
- The SYSCON block controls use of the CLKOUT pin, which must also be configured through IOCON (see [Section 4.3](#)).

4.2.1 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 system 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 192 MHz FRO.
 - FRO 32 MHz output (fro_32m) from internal oscillator.
 - The 32 MHz crystal oscillator output clock (clk_in).
 - FRO 1 MHz output (fro_1m) from internal low-power oscillator.
 - FRO 24 MHz output (fro_24m) from internal oscillator.
 - The 32.768 kHz crystal oscillator output clock or 32.768 kHz FRO clock (32k_osc).
2. See the section [Section 45.4.1](#).
3. Select the divider value for the system clock in the section [Section 45.4.1.1](#).
4. Enable the clock to the memories and peripherals used in the application.

4.2.2 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 determine the frequency of the watchdog oscillator, which varies over a wide range depending on process and temperature, accurately.

The clock frequency to be measured and the reference clock are selected in the input mux block (see the sections [Section 45.4.5.2.9](#), [Section 45.4.10.2.2](#), and [Section 45.4.5.2.10](#)).

Details on the accuracy and measurement process are described in [Section 45.4.10.2.2](#).

4.3 Pin description

Table 9. SYSCON pin description

Function	Type	Pin	Description	Reference
CLKOUT	O	PIO0_9, PIO0_18	Clock output	Section 15

4.4 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 10](#) describes signals on the clocking diagram. The device operates at a maximum CPU frequency of up to 32 MHz, and a maximum SPIFI operation of up to 64 MHz.

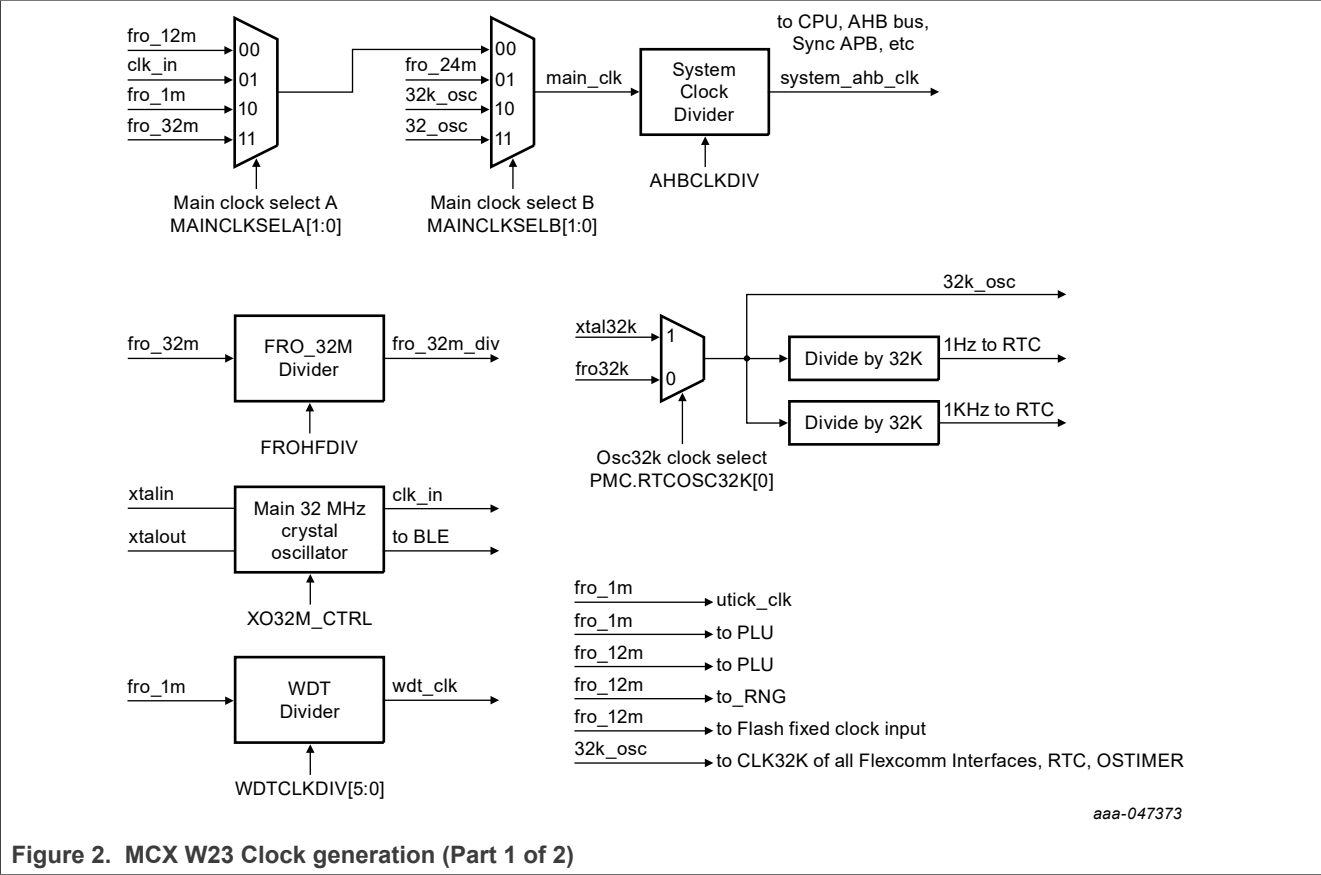
Note: The indicated clock multiplexers shown in [Figure 2](#) are synchronized. To operate, the selected clock must be running and the targeted clock must also be running, so the multiplexer can gracefully switch between the two clocks without glitches. Other clock multiplexers are not synchronized. If a glitch-free output is required, the output divider can be stopped and restarted gracefully during switching.

The low-power oscillator provides a frequency in the range of 1 MHz. The accuracy of this clock is limited to $\pm 25\%$ 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 4.2.2](#)).

Note: The maximum allowed frequency for main clock and system clock (to CPU0, AHB, Sync, and so on) is 32 MHz (see [Figure 2](#)).

Table 10. Clocking diagram signal name descriptions

Name	Description
32k_osc	The 32.768 kHz clock source. It is selected as either FRO32K or XTAL32K in the RTCOSCCTRL register.
clk_in	The internal clock that comes from the 32 MHz crystal oscillator
frg_clk	The output of each fractional rate generator (FRG) to the Flexcomm clock. Figure 3 shows each FRG and its source selection.
fro_12m	12 MHz divided down from the on-chip 192 MHz FRO.
fro_32m	The divided down high-speed output of 32 MHz from the on-chip 192 MHz FRO.
fro_32m_div	The output of the fro_32m divider.
fro_24m	24 MHz divided down from the on-chip 192 MHz FRO.
fro_48m	48 MHz divided down from the on-chip 192 MHz FRO. This 48 MHz clock signal is suitable for use by the SPIFI.
fro_64m	64 MHz divided down from the on-chip 192 MHz FRO. This 64 MHz clock signal is suitable for use by the SPIFI.
main_clk	The main clock that is the source for the system clock used by the CPU and AHB, memories, and potentially many others. The main clock and its source selection are shown in Figure 2 .
system_ahb_clk	The system clock used by the CPU and AHB, memories, and potentially many others.
fro_1m	The output of the 1MHz 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.



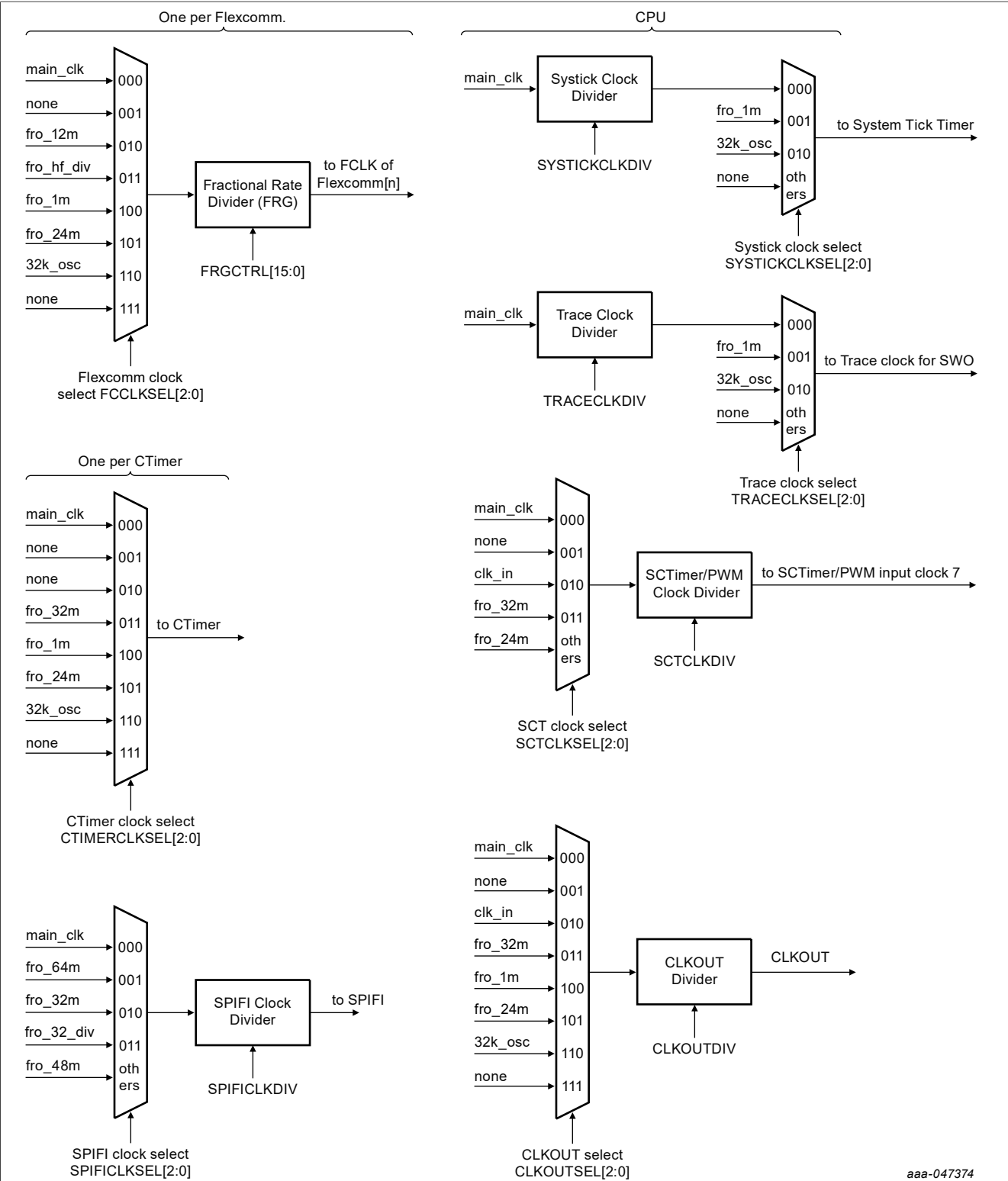


Figure 3. MCX W23 Clock generation (Part 2 of 2)

4.5 Functional description

4.5.1 Reset

Reset has the following sources:

- RESET_N pin
- Watchdog reset
- Power-on reset (POR)
- Brownout detect (BOD)
- Arm system reset
- ISP-AP debug reset
- Software reset

After power-on reset or upon the BOD reset, when the operating voltage has attained a usable level, the FRO_192M is started. After the FRO-start-up time, the FRO_192M 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.
3. The boot code in the ROM starts. The boot code performs the boot tasks and may jump to the flash when done.

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

4.5.2 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. So, it is necessary to pay attention during clock switching.

Glitch-free multiplexing of unaligned clocks is implemented in the selection of:

- main clk (the clock on which the CPU is running)
- in the power management module switching between 32 kHz, 1 MHz, 12 MHz

In general, for the peripheral modules, clock selection can glitch. Synchronization is not required because of the way the selection of their clocks must be configured:

1. Make sure that the module is held in reset
2. Configure the clock (since the module is held in reset, glitches that may occur during clock switching are not harmful).
3. Release the module reset

To provide a glitch free output, all the dividers can be halted and restarted during clock switching.

4.5.3 Brownout detection

To monitor the voltage of the internal VBAT, this device incorporates two brownout detectors. If the voltage drops to below one of the selected voltages (see the sections BOD1 and BOD2 in the Register description ([Ref.](#)

5). the BODs assert an interrupt to the NVIC or issues a reset (see the section Reset control (RESETCTRL) in the Register description [Ref. 5](#)).

The interrupt signal can be enabled in the interrupt enable register in the NVIC to cause a CPU interrupt (see [Section 3.2.1](#)). If not, software can monitor the signal by reading a dedicated status register.

If one of both of the BODs and their interrupts are enabled in a reduced power mode, a BOD interrupt can wake up the chip from that reduced power mode.

If one or both of the BODs are enabled in a reduced power mode and set up to cause a reset, the forced BOD reset can wake up and reset the chip from that reduced power mode. Upon a forced BOD reset, the PMC is reset. This causes the RESETCTRL register to be cleared which disables the reset function of the BODs. The BOD1 and BOD2 registers in the PMC are not reset and will keep their values upon a forced BOD reset.

4.5.4 Start-up behavior

The FRO 12 MHz oscillator division from the FRO_192M provides the default clock at reset. It also provides a clean system clock shortly after the supply pins reach operating voltage (for details of start-up timing, see the MCX W23 data sheet; [Ref. 7](#)).

Note: The ROM boot code may switch to a higher frequency (for example, 16 MHz, 24 MHz, or 32 MHz) depending on the settings in the flash protected area (IFR).

4.5.5 Flash accelerator functional description

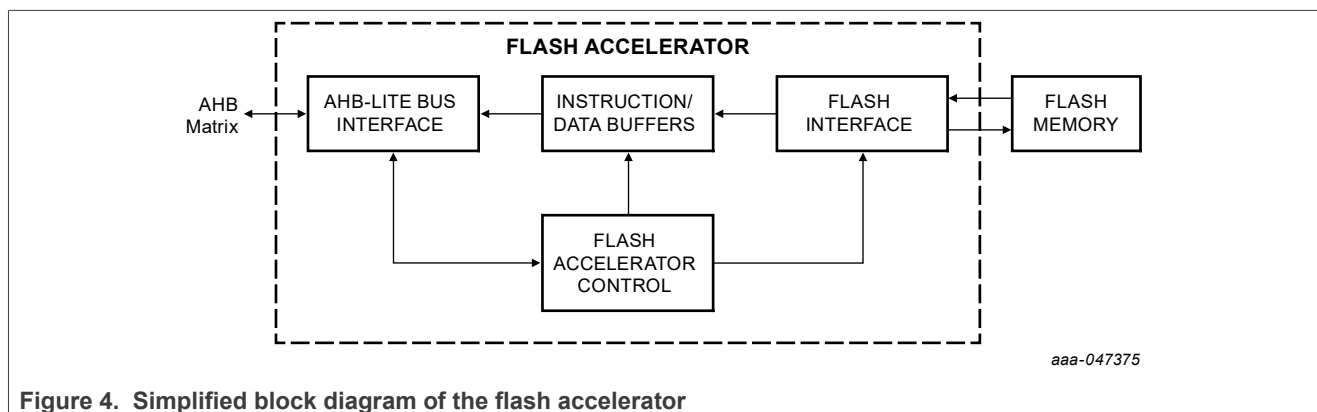
The flash accelerator is also known as the flash memory controller (FMC). The FMC is distinct from and interfaces with the flash controller.

The flash accelerator block maximizes performance of the CPU (when it is running code from flash memory) and helps to reduce power consumption (see the section "Flash memory unit (FMU)" in the Register description [Ref. 5](#)).

The flash accelerator is divided into several functional blocks:

- AHB matrix interface
All bus masters that have a connection to the matrix slave port used for flash memory, can access it.
- 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.



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

4.5.5.1 Flash memory bank

The flash accelerator does not control flash programming operations. They 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.

4.5.5.2 Flash programming constraints

The flash memory does not allow accesses during programming and erase operations. So, the flash accelerator must 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 can result in a watchdog timeout. The user must be aware of this possibility and take steps to ensure 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 that stale data are 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 causes 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 32 MHz.*

4.5.6 Frequency measure function

The frequency measure circuit is based on two 14-bit counters, one that the reference clock clocks and one that the target clock clocks. Synchronization between the clocks is performed at the start and at the end of each count sequence.

Software setting a control/status bit in the "Frequency measure function control register (FREQ_ME_CTRL)" (see the Register description [Ref. 5](#)) initiates a measurement cycle. The software can then poll this same measurement-in-progress bit which the hardware clears when the measurement operation is completed.

When the reference counter rolls over, the measurement cycle terminates. At that point, the state of the target counter is loaded into a capture field in the FREQMEAS register and the measure-in-progress bit is cleared. Software can read this capture value and apply a specific calculation to it which returns the precise frequency of the target clock in MHz.

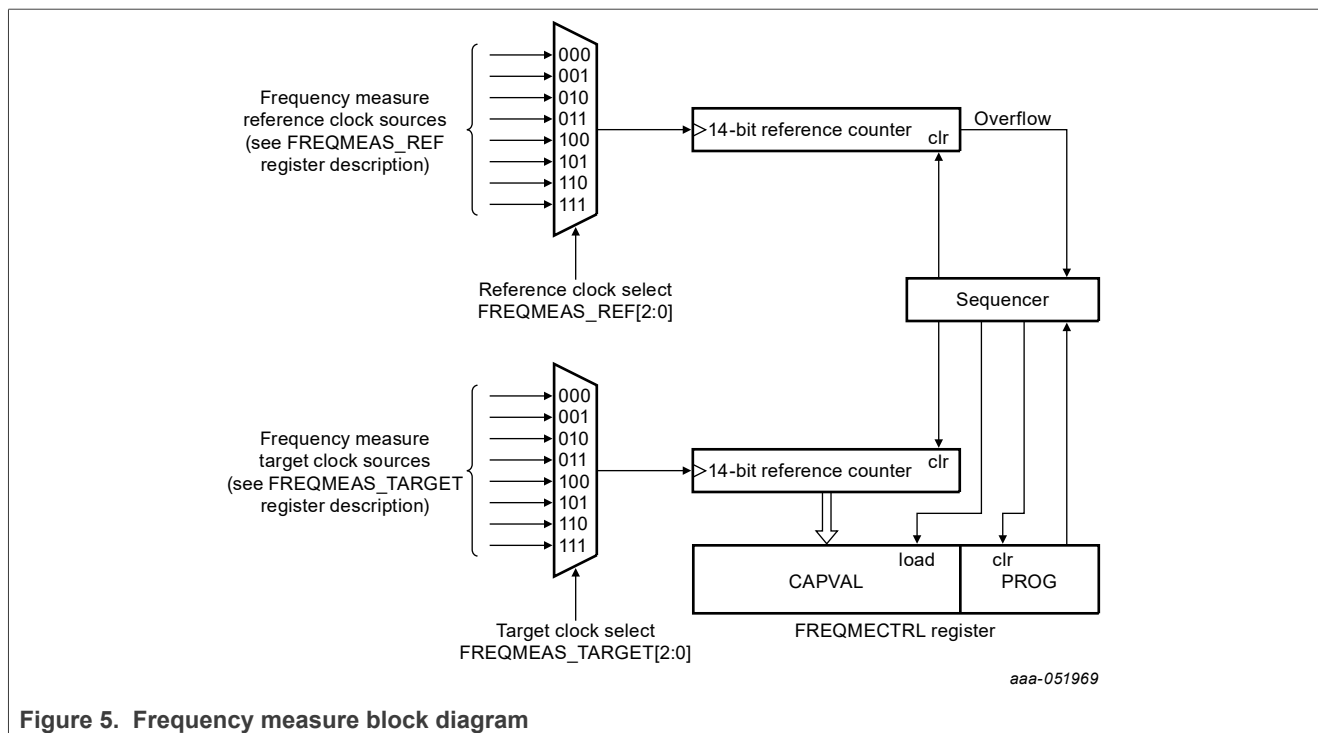


Figure 5. Frequency measure block diagram

Figure 5 shows a block diagram of the frequency measure function. Also see:

- [Section 4.2.2](#), Measure the frequency of a clock signal
- Frequency measure function control register (FREQ_ME_CTRL)(see the Register description [Ref. 5](#)).
- Selection for frequency measurement reference clock (FREQMEAS_REF)(see the Register description [Ref. 5](#)).
- Selection for frequency measurement target clock (FREQMEAS_TARGET)(see the Register description [Ref. 5](#)).

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 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 can 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) adds to the measurement error of the target clock. In general, this additional error is less than the uncertainty of the reference clock.

If the reference frequency exceeds the target frequency by a very large margin (25x or more), a modest loss of accuracy can also occur. Accuracy is not a simple function of the magnitude of the frequency difference. Nearly identical frequency combinations, still with a spread of about 43x, result in errors of less than 0.05 %.

If the target clocks and the reference clocks are different by more than a factor of approximately 500, the accuracy decreases to ± 4 %.

5 Flash

5.1 General description

This chapter describes the flash controller which is part of the MCX W23 device.

5.2 Features

- A 1-MByte block of main flash memory for vector space, code, and data storage
- 32 KByte of non-volatile protected flash region (PFR) separate from the main flash memory
- 512 bytes of program buffer space for flash programming
- Analog delay block to manage self-timed read operation
- Transparent AHB read and write ports
- Read port designed as an interface to the flash cache, single entry, and prefetch buffers
- APB control and status registers interface
- Auto-initialization after reset
- Error correction code (ECC) management, including single-bit correction and double-bit fault detection

5.3 Block diagram

[Figure 6](#) shows a functional block diagram of the controller. For clarity, some connections between blocks are not presented. The controller top level instantiates the hard block (the flash with the analog delay block) and blocks that contain all logic. The logic blocks are subdivided into subblocks for each of the different clock domains and an additional block that manages all clocks and resets.

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

Flash operations (erase, program) and reading a single word can be performed for CPU frequencies of up to 32 MHz. For frequencies of 20 MHz and higher, a read wait state is required. For CPU frequencies below 20 MHz, zero read wait states are required. Programming the FMU.FCTRL[RWSC] register field calling the flash_init() function can set the number of read wait states.

Note: When performing AHB reads of the flash memory contents, a hardware interrupt occurs if an unrecoverable error is detected.

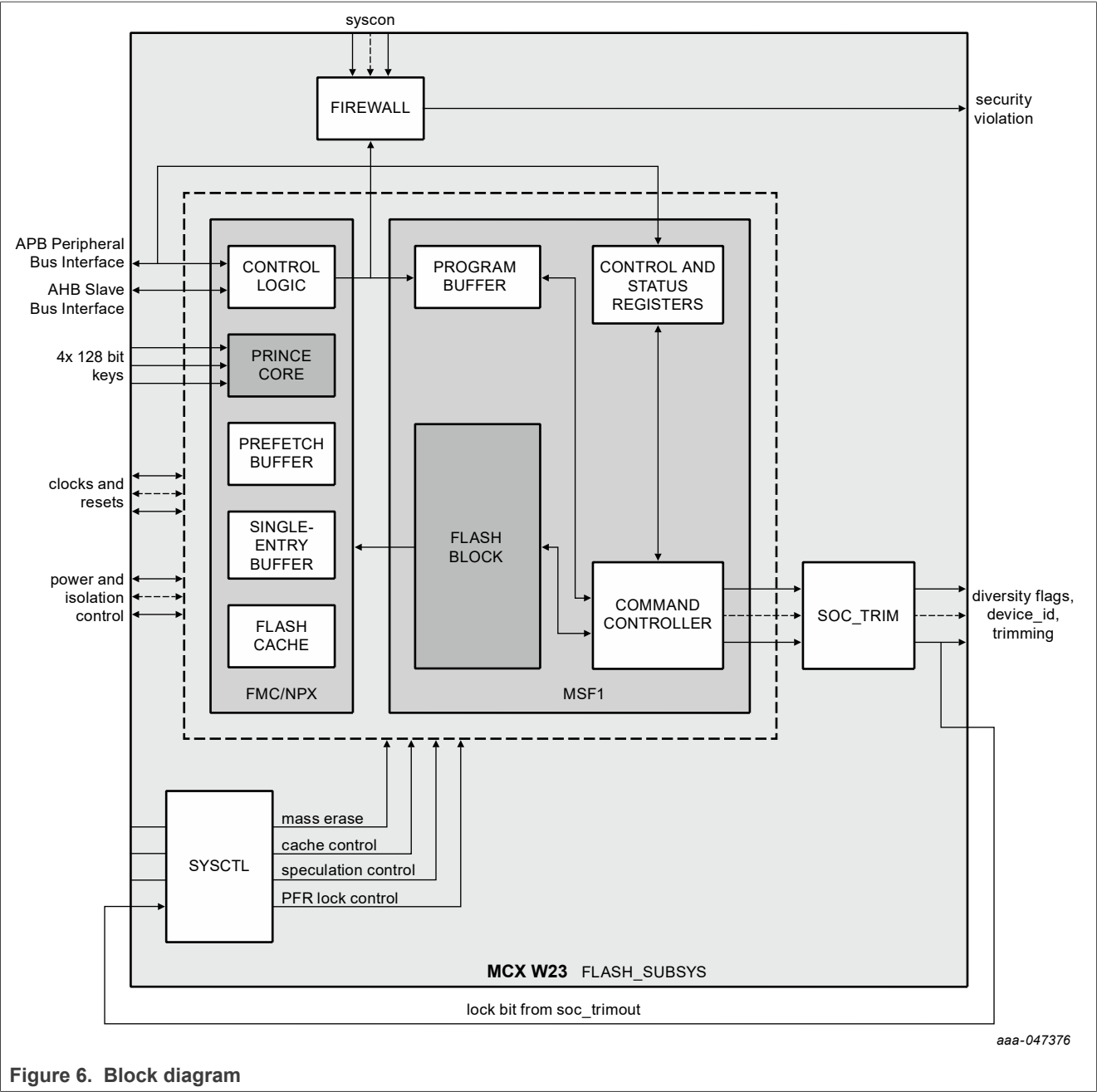


Figure 6. Block diagram

5.4 Usage

The flash can be accessed from a host using bootloader commands in ISP mode or from an application using IAP (see [Section 8](#)). Some additional functions are available in the flash driver.

5.4.1 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 for the exact locations and its 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.

5.4.2 Erasing

The flash cannot be accessed while an erase or write command is executed. It means that code in flash cannot be executed and data from flash cannot be read while erasing or writing. The functions that erase or write to flash are part of the ROM, so they can be safely executed. To avoid that interrupts are executed from flash, interrupts must be disabled or the vector table, the interrupt handlers, and all code and data they use must be in RAM.

Erasing must be done per sector. So, the start address and the size of an erase operation must be a multiple of 8192 bytes. In ISP mode, FlashEraseAll (see [Section 8.5.4](#)) and FlashEraseRegion (see [Section 8.5.5](#)) can be used. In IAP mode, flash_erase() (see [flash_erase](#)) can be used. After erasing a sector, each phrase in that sector can be written once at most. If you want to write to (a part of) a phrase that has already been written, you first must erase the entire sector containing the phrase. Then you must write the entire phrase again. Phrases that are not written cannot be read using the ISP command. It returns an error. They read as 0xFF in IAP mode.

5.4.3 Programming

Programming can be done per phrase (start address and size a multiple of 16 bytes). In ISP mode, WriteMemory (see [Section 8.5.7](#)) can be used. When the size is not a multiple of 16, this command does not give an error. It programs the extra bytes as 0 until the next phrase boundary. In IAP mode, flash_program (see [Section 9.2.2.3](#)) can be used. After an erase, each phrase can only be written once.

5.4.4 Reading

The flash can be read in ISP mode using the ReadMemory command (see [Section 8.5.6](#)). If the area read with this command overlaps with a flash phrase that has not been written after an erase, the command returns an error. In IAP mode, the flash can be read using the FLASH_Read() driver function or by reading from the flash addresses directly (memory-mapped). Phrases that are not written after an erase read as 0xFF. The FLASH_Read() driver function also uses memory-mapped access to the flash.

5.4.5 Bit errors

An error correcting code (ECC) that adds 9 bits that cannot be accessed protects each flash phrase. These additional ECC bits allow the flash to correct 1 bit error (of the 137 bits in a phrase) transparently. They also allow the detection of 2 bit errors (of the 137 bits in a phrase). If that is the case, a bus error (leading to a HardFault exception) is generated. If more than 2 bit errors are present in the 137 bits of a phrase, incorrect data may be returned without indication that it is incorrect.

5.4.6 Caching

Additional configuration of the flash accelerator (see [Section 4.5.5](#)) is available. The flash has a read cache that can be cleared by calling the FLASH_CacheClear() driver function. When reading from the flash, it already starts reading the next bytes as well. This behavior can be modified by calling the FLASH_CacheSpeculationControl() driver function. If the system clock frequency is changed, the flash may require additional wait states for reading. These wait states can be configured by calling the CLOCK_SetFLASHAccessCyclesForFreq() driver function. This function must be called before increasing the system clock frequency or after decreasing the system clock frequency.

6 Boot ROM

This chapter applies to all MCX W23 parts.

6.1 Features

The MCX W23 parts contain 256 kB on-chip ROM with:

- Bootloader that allows various boot options and APIs:
 - Based on ISP pin or CMPA setting in PFR region (see [Section 10](#)); supports automated booting from internal flash
 - IAP calls (see [Section 8](#) and [Section 5](#))
 - Supports SPI flash recovery boot from 1-bit SPI flash device (see [Section 6.3.3](#) for more details)
- Part of the Mbed TLS crypto lib

6.2 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 the (deep) power-down state or the power-off state.

The MCX W23 has internal flash for code and data storage. Images are stored in internal flash. Upon boot, the image is validated and the boot ROM vectors point to on-chip flash.

Depending on the values of the CMPA bits, the ISP pin, and the image header type definition, the bootloader decides if it boots from internal flash or to run in ISP mode (see [Section 6.4](#)). To determine the boot source, the MCX W23 reads the status of the ISP pin (see [Table 11](#)).

Table 11. Boot mode and ISP download modes based on ISP pin

Boot mode	ISP0 (PIO0_20 pin)	Description
passive boot	HIGH	The MCX W23 looks for a valid image in the internal flash. If no valid image is found, the MCX W23 checks if the SPI flash recovery boot is enabled. If not, it enters ISP boot mode based on DEFAULT_ISP_MODE bits defined in Table 12 .
ISP boot	LOW	The bootloader enters ISP boot mode based on DEFAULT_ISP_MODE bits defined in Table 12 . One of the serial interfaces (UART2, I ² C1, SPI0) is used to download an image from the host to the internal flash. The first valid ping packet on USART, I ² C, or SPI locks ISP in that interface. Note: If DEFAULT_ISP_MODE is set to disable ISP and the boot pin is LOW, the bootloader does not respond to commands and does not try to boot a valid image in the internal flash.

Table 12. ISP download mode based on DEFAULT_ISP_MODE bits (BOOT_CFG[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 MCX W23 detects the active peripheral from one of below serial interfaces: UART2, I ² C1, SPI0.
disable ISP	0	0	1	disable ISP mode
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.

Table 12. ISP download mode based on DEFAULT_ISP_MODE bits (BOOT_CFG[6:4], word 0 in CMPA)...continued

ISP boot mode	ISP_MODE_2	ISP_MODE_1	ISP_MODE_0	Description
disable ISP	1	0	1	disable ISP mode
disable ISP	1	1	0	disable ISP mode
disable ISP	1	1	1	disable ISP mode

[Table 13](#) shows the ISP pin assignments and is the default pin assignment the ROM code uses. It cannot be changed.

Table 13. ISP pin assignments

ISP pin	Port pin assignment
ISP0	PIO0_20
USART ISP mode	
FC2_TXD	PIO0_19
FC2_RXD	PIO0_18
I²C ISP mode	
FC1_SDA	PIO0_13
FC1_SCL	PIO0_14
SPI ISP mode	
FC0_SCK	PIO0_4
FC0_SSEL0	PIO0_0
FC0_MISO	PIO0_3
FC0_MOSI	PIO0_2

[Figure 7](#) shows the top-level boot process. The boot starts after reset is released.

The CPU clock is 12 MHz based on the 12 MHz FRO. When the Cortex-M33 starts the bootloader, the SWD access is not enabled yet. So, the debugger cannot connect to the CPU during this period.

The bootloader completes initialization of the device and applies settings from the NMPA/CMPA and CFPA, potentially reconfiguring the system clock settings.

The boot ROM determines the boot mode based on the state of the ISP pin. After the boot mode is determined and if a valid image is present in internal flash, the bootloader validates the vector table and image header.

The boot ROM checks the following for image validity check:

- Validate image using image header and CMPA settings when secure boot is enabled (see [Section 7](#) 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 the integrity check and the authentication check are not enabled.
- Validate the TZ-M image type if the previous checks passed.

The first words of the image follow the format mentioned in [Table 14](#). The bootloader begins scanning for a user image at address 0x0 in internal flash by examining the image type marker at 0x0000 0024. If the value matches any supported image type markers, then validation of an image header begins. After the validation of the image header is completed, the qualification continues by examining the TZ-M 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 14](#)). 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 makes up the CRC value field. It means that the CRC calculated skips the CRC value field. The result is then compared to the `offsetToSpecificHeader` entry in the structure and, if a match exists, the image is considered valid. Otherwise, the image is considered invalid. If the image is not a CRC image, CRC is not performed.

If it is a signed image, then the `imageLength` field value is used as the length to perform an authentication on. The authentication is 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 14. Image header for the MCX W23 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	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 any other value
0x24	4	imageType	image type 0x0000 – plain 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 offset to certificate block header if the image type is 0x0001 0x0004, or 0x8001 the <code>crcChecksum</code> if the image type is 0x0002 or 0x0005
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 any other value
0x38	8	vector table	Cortex-M33 vector table entries.

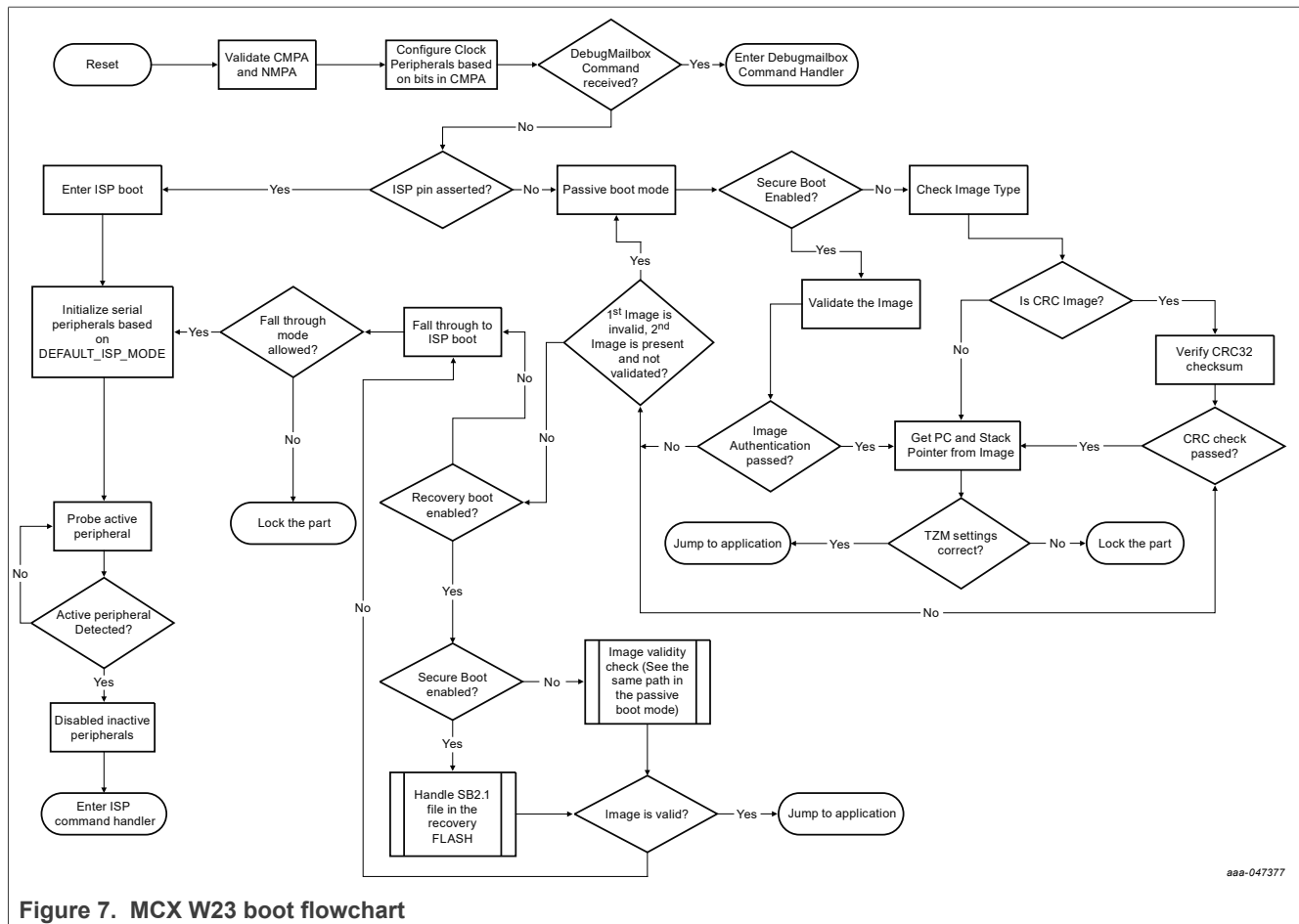


Figure 7. MCX W23 boot flowchart

There are several paths into the ISP mode:

- Entry through the ISP pin assertion
 - This mechanism is used on EVK boards and customer boards where switches and jumpers are affordable in terms of space and cost.
 - This method is used to recover a part programmed with a corrupted image which is not detectable by ROM. When in ISP mode, commands are provided to reprogram the flash with proper image.
- Entry through fall-through mode
 - When the user image in flash is blank or invalid, the ROM enters ISP mode.
 - This mode of entry allows programming factory fresh parts through an ISP interface.
- Entry through ROM API call
 - The application code uses this path to implement in-field updates using the ISP mechanism.
- Entry through the debug mailbox command
 - This mechanism is used to recover a part programmed with a corrupted image. For instance, onboards/ products where the ISP pin mechanism is not affordable and SWD pins are accessible. Or when ISP mode is disabled through the "default ISP mode" field in the BOOT_CFG configuration word of the CMPA.
 - For security-sensitive applications, an option to control access to this mechanism is provided through the debug authentication procedure.

6.3 Boot modes

The boot modes include:

- [Section 6.3.1](#)
- [Section 6.3.2](#)
- [Section 6.3.3](#)

6.3.1 Passive boot mode

The CPU clock is set to the boot speed specified in CMPA field. It boots directly from internal flash based on the image header (see [Table 14](#)) and the result of the image validation.

6.3.2 ISP boot mode

The ISP boot mode can be entered as a result of failed image verification or if ISP pin forces the device into ISP mode. The ISP mode is used for:

- Downloading the (initial or updated) image 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 [Section 8](#).

6.3.3 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 used 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 attempts to recover the device by booting/executing the SB2.1 file present on the external flash device.

For more details about the SB2.1 file format, see [Section 7.3.6](#).

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 (SPI_FLASH_CFG[4:0], at 0x104004) in protected flash to determine if SPI flash recovery is enabled. If so, the boot ROM tests SEC_BOOT_EN (SECURE_BOOT_CFG bits [31:30], at 0x104014) in protected flash.

If SEC_BOOT_EN is non-zero, the image present in an SB container on external flash can be programmed into internal flash or booted into internal SRAM. 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 MCX W23 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 COMxx get-property 12
```

For more details, see [Section 8.5.11.2](#) for more details.

6.4 PFR region definitions

The PFR region is used as the persistent storage for the secure boot and the IC-specific parameters. It starts at fixed address 0xFE000. For more details, see [Section 9](#) and [Section 10](#).

Table 15. Image header for the MCX W23 devices

Region	Field	Description
0x0FE000-0x103FFF	customer in-field programmable area (CFPA)	see Section 10.3
0x104000-0x107FFF	customer manufacturing/factory programmable area (CMPA) and key store area (KSA)	see Section 10.1 and Section 10.2
0x108000-0x109FFF	NXP manufacturing programmed area (NMPA)	Reserved for NXP internal use only (see Section 10.4).

6.5 MbedTLS ROM library

Part of the MbedTLS cryptolibrary, version 2.28.0, is available in the on-chip boot ROM.

On A0 silicon, version 2.27.0 is in the on-chip boot ROM.

[Table 16](#) lists all the modules of the mbedTLS library and indicates if they are in ROM or not.

Table 16. MbedTLS modules that are part of ROM

MbedTLS Module	In ROM?
aes	yes
aesni	yes
arc4	yes
aria	yes
asn1parse	yes
asn1write	yes
base64	yes
bignum	yes
blowfish	yes
camellia	yes
CCM	yes
chacha20	yes
chachapoly	yes
cipher	yes
cipher_wrap	yes
ctr_drbg	yes
debug	yes
des	yes
dhm	yes
ecdh	yes
ecdsa	yes

Table 16. MbedTLS modules that are part of ROM...continued

MbedTLS Module	In ROM?
ecjpake	yes
eCP	yes
eCP_curves	yes
entropy	yes
entropy_poll	yes
gcm	yes
havege	yes
hkdf	yes
hmac_drbg	yes
md	yes
md2	yes
md4	yes
md5	yes
memory_buffer_alloc	yes
net_sockets	yes
nist_kw	yes
oid	yes
padlock	yes
pem	yes
pk	yes
pk_wrap	yes
pkcs5	yes
pkcs12	yes
pkcs11	yes
pkparse	yes
pkwrite	yes
platform	yes
platform_util	yes
poly1305	yes
ripemd160	yes
rsa	yes
rsa_internal	yes
sha256	yes
sha512	yes
sha1	yes
threading	yes

Table 16. MbedTLS modules that are part of ROM...continued

MbedTLS Module	In ROM?
timing	yes
version	yes
version_features	yes
certs	no
cmac	no
error	no
mps_reader	no
mps_trace	no
psa_crypto	no
psa_crypto_aead	no
psa_crypto_cipher	no
psa_crypto_client	no
psa_crypto_driver_wrappers	no
psa_crypto_ecp	no
psa_crypto_hash	no
psa_crypto_mac	no
psa_crypto_rsa	no
psa_crypto_se	no
psa_crypto_slot_management	no
psa_crypto_storage	no
psa_its_file	no
ssl_cache	no
ssl_ciphersuites	no
ssl_cli	no
ssl_cookie	no
ssl_msg	no
ssl_srv	no
ssl_ticket	no
ssl_tls	no
ssl_tlsyes3_keys	no
x509	no
x509_create	no
x509_crl	no
x509_crt	no
x509_csr	no
x509write_crt	no

Table 16. MbedTLS modules that are part of ROM...continued

MbedTLS Module	In ROM?
x509write_csr	no
xtea	no

For some of the MbedTLS software algorithms, a MCX W23-specific port is available to make optimal use of the security hardware accelerators of MCX W23 (called MbedTLS KSDK port). Also these portable files are in ROM.

Table 17. MbedTLS KSDK modules that are part of ROM

MbedTLS KSDK module	In ROM?
ksdk_mbedtls.c	yes
ksdk_mbedtls.h	yes
ksdk_mbedtls_config.h	yes
sha1_alt.h	yes
sha256_alt.h	yes
des_alt.h	yes
aes_alt.h	yes
ecp_alt.h	yes

When using the MbedTLS ROM library in an application, the function `mbedtls_rom_init` must always be called before any other MbedTLS function can be called. This function initializes the data memory the MbedTLS ROM uses. The MbedTLS ROM uses the RAM address range 0x2000C000 until 0x2000D000. So, the rest of the application cannot use this memory section.

How to use MbedTLS ROM in the MCX W23 is demonstrated in various sample applications that are available in the MCX W23 SDK.

7 Secure boot ROM

This document describes the secure boot ROM architecture for MCX W23 series.

The secure part of ROM bootloader provides following basic operations:

- Secure boot
- Secure firmware update
- Security-related miscellaneous functions

The ROM bootloader provides an API that allows integration of bootloader operations into customer applications.

7.1 Function description

7.1.1 Secure boot

Secure boot prevents that unauthorized code is executed on a given product. It achieves this level of security by always executing the ROM of the device coming out of reset. This process 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, control is transferred to it. In this way, a chain of trusted code from the ROM to the user boot code is established. This chain can be further extended, through the verification of digital signatures associated with the boot code.

The boot code is signed with RSA private keys. In this architecture, RSA signatures over it are verified to attest its authenticity. The corresponding RSA public keys used for signature verification are contained in X.509 certificates that are part of the signed image. Root of trust keys sign those X.509 certificates. The ROM can check the authenticity of 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 it only performs CRC32 checking, depending on the configuration.

7.1.2 Secure firmware update

If firmware updates are performed in-the-field when secure boot is enabled, a secure firmware update mechanism is preferred. Otherwise, inauthentic firmware may be written to the device, causing it not to 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. Hiding the code 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 (UART, I2C, or SPI). Or it can be provided to the ROM API as a complete binary file (see [Section 6.3.3](#)).

7.1.3 Extending the chain of trust

When secure boot has transferred CPU control to user code, that code may have to load additional pieces of code. The loading of additional pieces of code establishes another link in the chain of trust. The process can continue for as many nested submodules as required, with each parent code module authenticating the next step in the chain.

To verify signatures on the additional code images, the ROM API is used from the customer code. 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.

7.1.4 Miscellaneous functions

The ROM supports various security-related additional functionalities. The main ones are:

- Initial boot state, as specified in Arm Platform Security Architecture Security Mode 1.0
- Loading of TrustZone-M preconfiguration during ROM secure boot
- Booting from encrypted internal flash regions using the PRINCE peripheral module
- Debug authentication
- Device provisioning (ROM embedded support for the initial secure provisioning of the boot keys)
- Part of the Mbed TLS cryptolibrary

7.1.5 Boot flow diagram

Settings written in the protected flash region (PFR) of internal device flash memory control the booting of the device. It is based on the ISP pin setting.

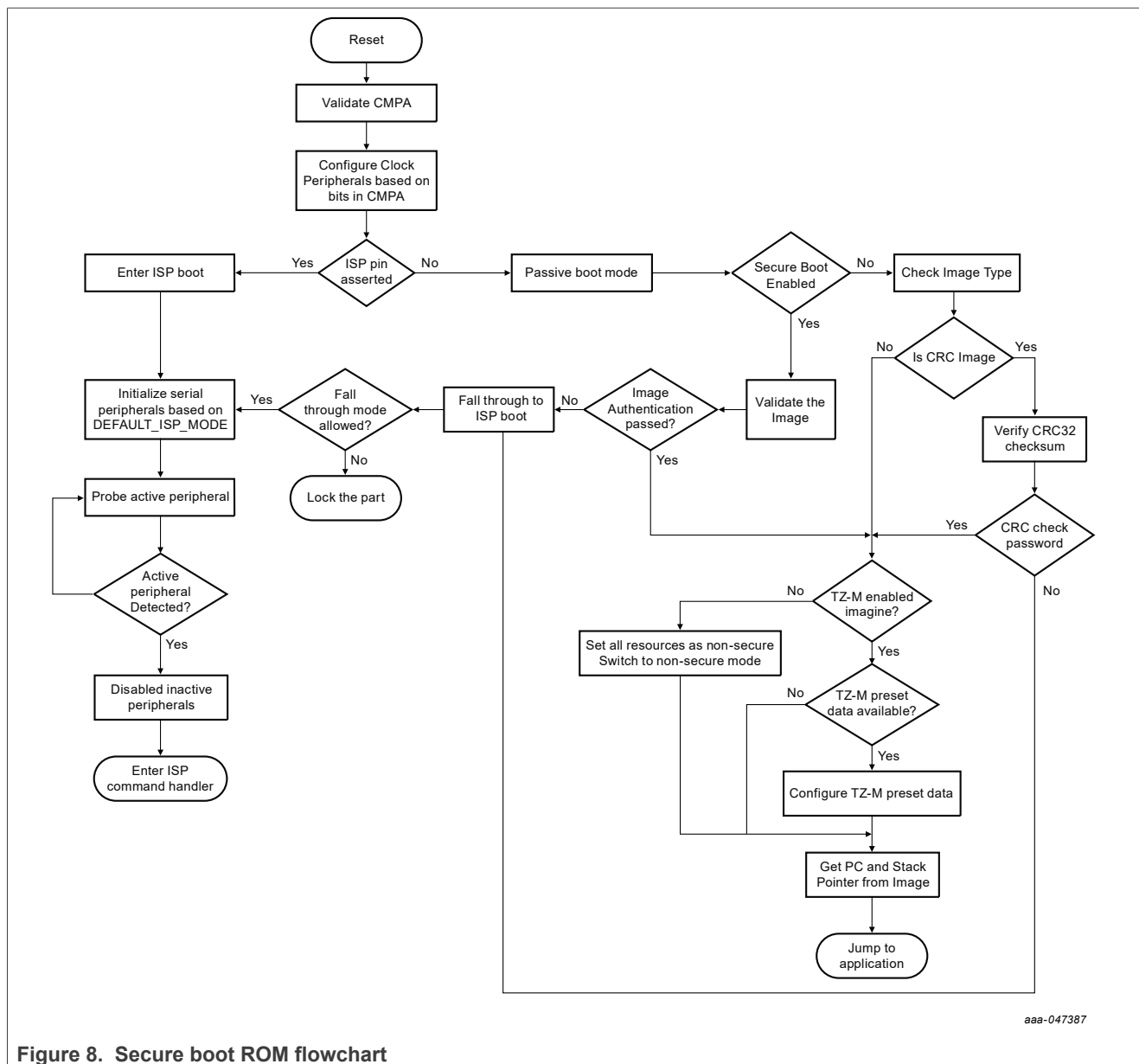


Figure 8. Secure boot ROM flowchart

7.2 Data structures

7.2.1 Overview

The MCX W23 stores configuration and PUF key store for the boot ROM in the protected flash region (PFR). It resides at the end of the flash region and can be programmed through ROM in ISP mode.

Note: The CMPA and the key store can only be programmed when the CMPA digest is entirely filled with 0. When the CMPA digest is written with a non-zero value, the ROM permanently seals the CMPA and the key store, making it impossible to change them again.

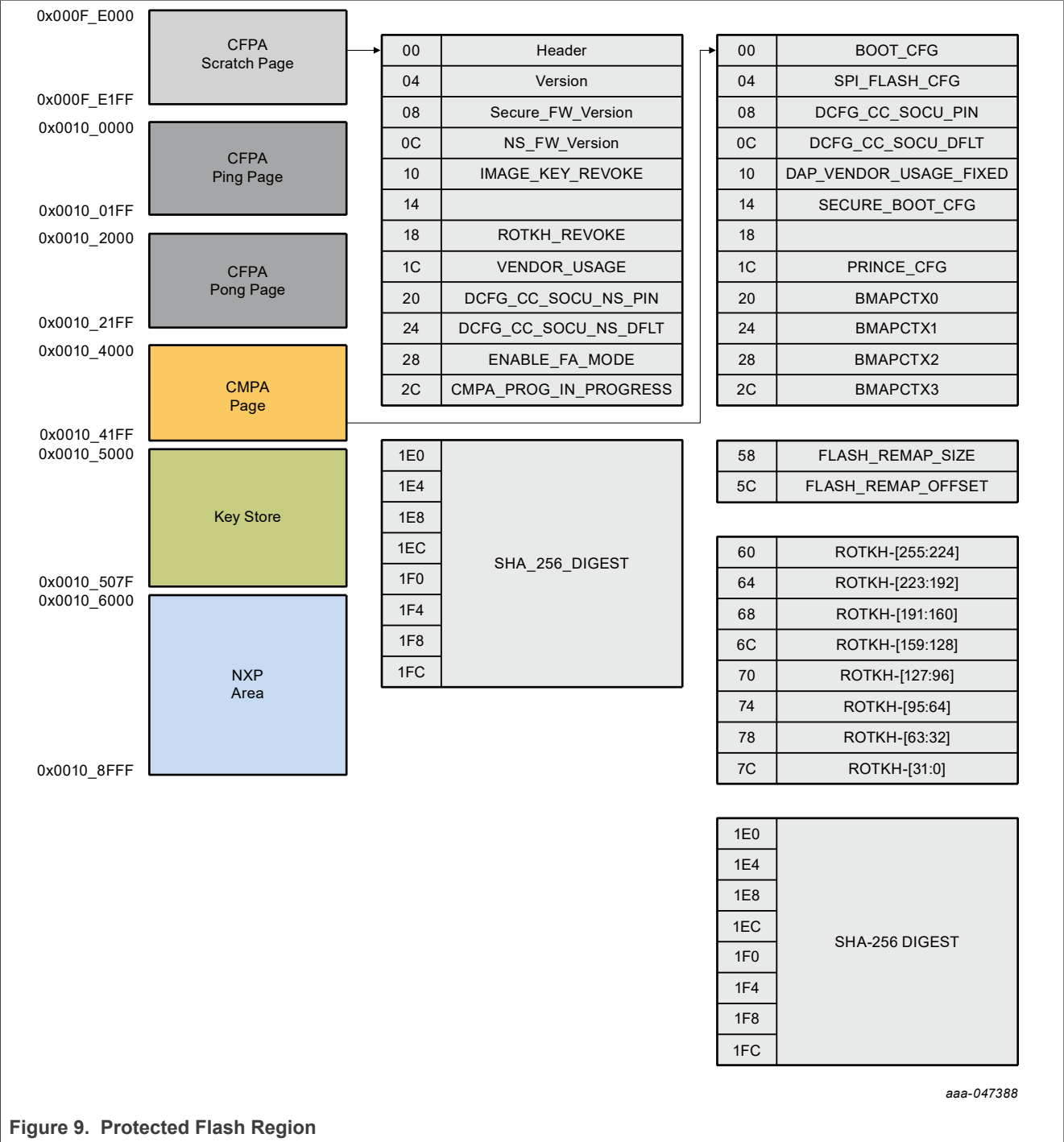


Figure 9. Protected Flash Region

7.2.2 Secure boot related configuration fields in PFR

7.2.2.1 Customer manufacturing/factory programmable area (CMPA) page

The customer manufacturing/factory programmable area (CMPA) page contains settings for a signed image in secure boot configuration, PRINCE configuration registers (if encrypted flash is required), and 32 bytes of root key table hash (RKTH). Only secure-boot-related fields are described in this chapter. Other fields are described in [Section 10.1](#).

Note: If the CMPA digest that is written is not entirely filled with 0, the ROM computes the digest and programs the computed digest as part of the CMPA. This action permanently seals the CMPA and the customer key store, making it impossible to change them again later. For other implications of writing the CMPA digest, see [Section 10.5](#).

Table 18. CMPA secure boot configuration overview

Address	Bytes	Name	Description
0x104014	4	SECURE_BOOT_CFG	Secure boot configuration flags
0x10401C	4	PRINCE_CFG	PRINCE configuration
0x104020	4	BMAPCTX0	Bitmap control for Prince memory context region 0, subregion (8 kB) enable bit
0x104024	4	BMAPCTX1	Bitmap control for PRINCE memory context Region 1, subregion (8 kB) enable bit
0x104028	4	BMAPCTX2	Bitmap control for PRINCE memory context region 2, subregion (8 kB) enable bit
0x10402C	4	BMAPCTX3	Bitmap control for PRINCE memory context region 3, subregion (8 kB) enable bit
0x104060	32	ROTKH	Root of trust keys table hash

PRINCE region base addresses are shown below:

- PRINCE region 0: 0x0000 0000 - 0x0003 FFFF
- PRINCE region 1: 0x0004 0000 - 0x0007 FFFF
- PRINCE region 2: 0x0008 0000 - 0x000B FFFF
- PRINCE region 3: 0x000C 0000 - 0x000F FFFF (0x000F E000 - 0x000F FFFF is reserved for PFR and cannot be encrypted using PRINCE)

SECURE_BOOT_CFG configuration word

Table 19. SECURE_BOOT_CFG word bit field definitions

Address	Bits	Name	Description
0x104014	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 reserved	Include NXP area in DICE CDI computation: 2'b00: not included 2'b01: included 2'b10: included 2'b11: included Reserved, filled with zeros.

Table 19. SECURE_BOOT_CFG word bit field definitions...continued

Address	Bits	Name	Description
	5:4	DICE_CUST_CFG reserved	Include CFPA page and key store area in DICE CDI computation: 2'b00: not included 2'b01: included 2'b10: included 2'b11: included Reserved, filled with zeros.
	7:6	SKIP_DICE reserved	Skip DICE CDI computation: 2'b00: Enable DICE 2'b01: Disable DICE 2'b10: Disable DICE 2'b11: Disable DICE Reserved, filled with ones.
	9:8	TZM_IMAGE_TYPE	TrustZone-M image mode: 2'b00: TZ-M image mode is taken from application image header 2'b01: TZ-M disabled image, boots to non-secure mode 2'b10: TZ-M enabled image, boots to secure mode 2'b11: TZ-M enabled image with TZ-M preset, boot to secure mode TZ-M pre-configured by data from application image header
	11:10	BLOCK_SET_KEY	Block PUF key code generation: 2'b00: Allow PUF key code generation 2'b01: Disable PUF key code generation 2'b10: Disable PUF key code generation 2'b11: Disable PUF key code generation
	13:12	RESERVED	Reserved, filled with zeros.
	15:14	DICE_INC_SEC_EPOCH	Include security epoch area in DICE CDI 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: 2'b00: enable BOOT_SEED 2'b01: disable BOOT_SEED 2'b10: disable BOOT_SEED 2'b11: disable BOOT_SEED
	19:18	BOOT_SEED_INC_NXP_CFG	Include NXP area in BOOT_SEED computation: 2'b00: not included 2'b01: included 2'b10: included 2'b11: included
	21:20	BOOT_SEED_CUST_CFG	Include CMPA area in BOOT_SEED computation: 2'b00: not included 2'b01: included 2'b10: included

Table 19. SECURE_BOOT_CFG word bit field definitions...continued

Address	Bits	Name	Description
			2'b11: included
	23:22	BOOT_SEED_INC_EPOCH	Include security epoch area in BOOT_SEED computation. See security epoch PFR fields below this table. 2'b00: not included 2'b01: included 2'b10: included 2'b11: included
	29:24	reserved	Reserved, filled with zeros.
	31:30	SEC_BOOT_EN	Secure boot enable: 2'b00: Plain image allowed (internal flash, with or without CRC) 2'b01: Boot signed images only (internal flash, RSA signed) 2'b10: Boot signed images only (internal flash, RSA signed) 2'b11: Boot signed images only (internal flash, RSA signed)

Security epoch consists of the following CFPA fields:

- IMAGE_KEY_REVOKE
- RKTH_REVOKE
- VENDOR_USAGE
- DCFG_CC_SOCU_NS_PIN
- DCFG_CC_SOCU_NS_DFLT

PRINCE_CFG

Contains various configurations for the PRINCE peripheral that the ROM bootloader has to be set during device start-up.

Table 20. PRINCE configuration

Address	Bits	Name	Description
0x10401C	0:0	reserved	reserved; filled with zeros.
	1:1	GEE	Global encryption enable
	4:2	reserved	reserved; filled with zeros
	5:5	GDE	Global decryption enable
	8:6	reserved	reserved; filled with zeros
	9:9	SLK	System lock enable The Prince APB system accesses to the control register or to any region-configuration register are ignored when this bit is set.
	12:10	reserved	reserved; filled with zeros
	13:13	GLK	Global lock enable The Prince APB system accesses to the control register or to any region-configuration register are ignored when this bit is set.

Table 20. PRINCE configuration...continued

Address	Bits	Name	Description
	31:14	reserved	reserved; filled with zeros

BMAPCTXx

When on-the-fly encryption/decryption of internal flash using PRINCE is enabled, ROM configures subregion-enable bits for a given memory region, according to value stored in this word.

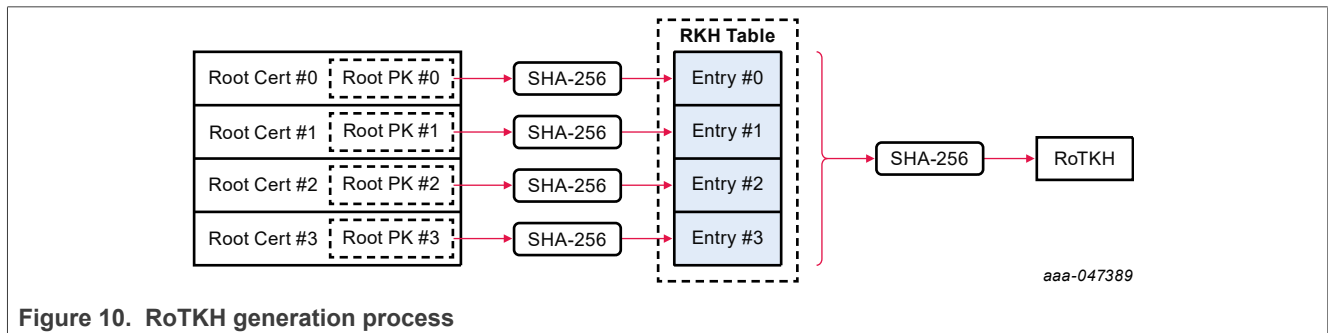
Table 21. PRINCE subregion enable bits

Address	Bits	Name	Description
0x104020	31:0	BMAPCTXx	Each bit in this field enables a subregion of crypto region x at offset 8kB*n, where n is the bit number. A 0 in bit n means encryption and decryption of data associated with subregion n is disabled. A 1 in bit n means that data written to subregion n during flash programming when GEE = 1 is encrypted. And flash reads from subregion n are decrypted using the PRINCE when GDE = 1. Note: In BMAPCTX3, bit 31 must be 0 because the boot ROM has reserved this subregion for PFR.

RoTKH

RoTKH is 32-byte SHA-256 hash of SHA-256 hashes of up to four root of trust public keys. Multiple root of trust public keys are supported, such that root of trust key revocation is possible.

The structure of this table is shown here:



Each entry in the table is an SHA-256 digest result, computed over the concatenation of the modulus and exponent (modulus || exponent, where the || symbol indicates a concatenation of data) of an RSA public key. Both modulus and exponent must be in big-endian (BE) byte order with the minimum number of bytes required to represent the value. For instance, a 3-byte value of [01 00 01] represents an exponent of 65537, while a single byte of that value represents an exponent of 3. The entire RKH table is itself hashed with SHA-. This final hash digest result is stored in the RoTKH 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 ROTKH = SHA256(RKH 0 || RKH 1 || RKH 2 || RKH 3)
```

The number of hash digests of keys in the RKH table must be from at least 1 through to a maximum of 4. Unused table entries must be set to all 0 bytes. When searching the RKH table for a hash digest of a key, the bootloader stops at the first entry that is all zeroes.

The extra root of trust public keys and root of trust certificates must be created in advance and held in reserve, should a root of trust public key have to be revoked. The customer is responsible for implementing the mechanism to determine if a root of trust key must be revoked and to set the appropriate RKTH_REVOKE bits. It is done through an authenticated connection with a server during a firmware update.

Note: Only one of the root of trust certificates whose keys are listed in the RKH table may be included in the certificate table of a signed image at a time.

Table 22. RKTH layout in CMPA

Address	Description	Address	Description
0x104060	RoTKH [255:224]	0x104070	RoTKH [127:96]
0x104064	RoTKH [223:192]	0x104074	RoTKH [95:64]
0x104068	RoTKH [191:160]	0x104078	RoTKH [63:32]
0x10406C	RoTKH [159:128]	0x10407C	RoTKH [31:0]

7.2.2.2 Customer field programmable area (CFPA) page

The customer field programmable area (CFPA) page contains three monotonic counters and RKTH revocation fields. Only secure-boot-related fields are described in this chapter. Other fields are described in [Section 10.3](#).

Table 23. CFPA page layout

Address	Bytes	Name	Description
0xFE008	4	Secure_FW_Version	Secure firmware version (monotonic counter)
0xFE00C	4	NS_FW_Version	Non-secure firmware version (monotonic counter)
0xFE010	4	IMAGE_KEY_REVOKE	Image key revocation ID (monotonic counter)
0xFE018	1	RKTH_REVOKE	Used for revocation of individual Root of Trust keys

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 is 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 X.509 serial number field in the image signing certificate is used in the following way: byte 0 is 0x3c, byte 1 is 0xc3. Byte 2 and byte 3 together 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 must be set on every revocation starting from lower bit 0 to 15:

0b0 → 0b1 → 0b11 → 0b111.....

To avoid damaging the device if power loss happens after an FW update, but before IMAGE_KEY_REVOKE is updated, the MCX W23 boot ROM allows a roll forward (only by 1). It 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 are rejected during the authentication process and fail to boot if SEC_BOOT_EN is set to boot only signed images.

Table 24. RKTH_REVOKE table bit field description

Address	Bits	Name	Description
0xFE018	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 is enabled 2'b10: RoT key3 is revoked 2'b11: RoT key 3 is revoked
	31:8	reserved	must be filled with zeros

7.2.3 Key storage in the protected flash region

MCX W23 uses the PUF controller for key wrapping. The PUF activation code and 8 key codes are stored in the PFR. The ROM manages and uses them during the boot and SB file processing. The PUF activation code and most of the key codes are stored in the NXP area. The user can store 2 key codes, created using the key provisioning process in the key store, using the `write_key_nonvolatile blhost` command. The content of key storage is also available to user application by using the `ffr_keystore_get_ac()` and `ffr_keystore_get_kc()` ROM API functions. For more information, see [Section 9.2.1](#). During start-up, 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. The ROM issues the PUF start procedure, which initializes PUF and loads the activation code, so that each key can be used if necessary.

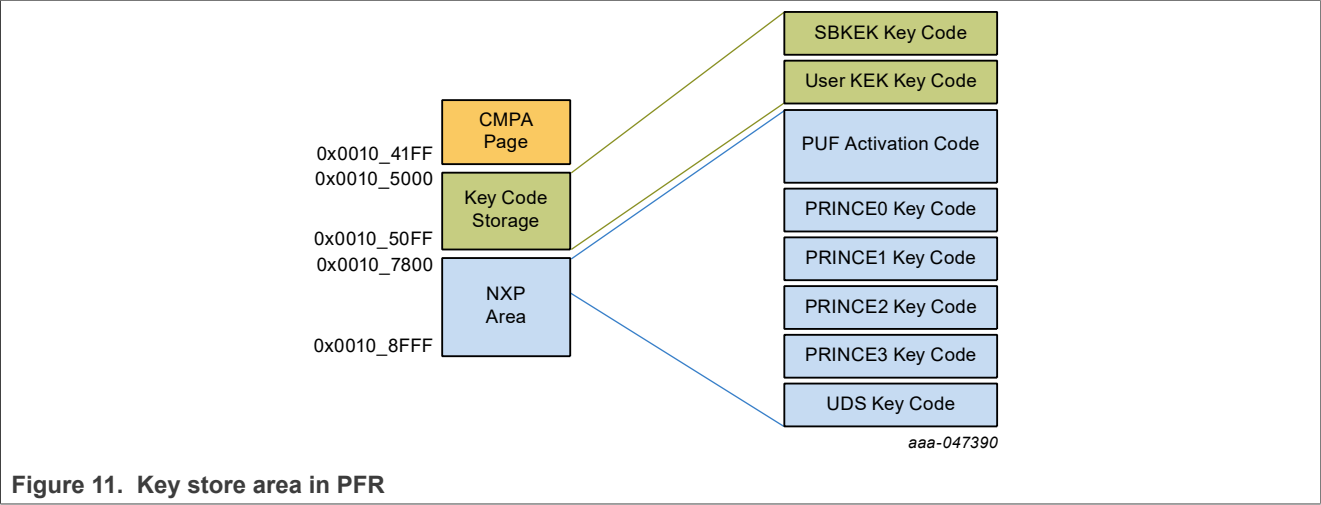


Figure 11. Key store area in PFR

Table 25. PUF key code storage area structure

Address	Size (bytes)	Name	Description
0x107800	4	key store header	marker A value of 0x95959595 means that activation code is valid.
0x107804	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.
0x107808	1192	activation code	device-specific PUF activation code generated by enroll command during key provisioning
0x105014	56	SBKEK key code	key code for wrapped SBKEK key
0x10504C	56	USERKEK key code	key code for wrapped USERKEK key
0x107D94	56	UDS key code	key code for wrapped UDS key
0x107CB4	56	PRINCE region 0 key code	key code for wrapped PRINCE region 0 key
0x107CEC	56	PRINCE region 1 key code	key code for wrapped PRINCE region 1 key
0x107D24	56	PRINCE region 2 key code	key code for wrapped PRINCE region 2 key
0x107D5C	56	PRINCE region 3 key code	key code for wrapped PRINCE region 3 key
0x107DCC	56	AES key code	key code for wrapped AES key

7.3 Keys

This chapter describes the purpose of individual keys used by the bootloader. The PUF generates (/wraps) key codes for given keys using key type and key length (/key value) parameters that blhost key provisioning commands supply. Keys stored with PUF key index 0 can only be unwrapped on a secure key hardware bus. So, only security peripherals (HASHCRYPT and PRINCE) connected to this bus can use this code. Depending on the value of "Enable HW user mode keys", which the bit in Image Type (word at offset 0x24) in the header of an image specifies, access to this secure key bus from a non-secure method can be restricted. When this bit is

set to 0, PUF can only access the keys are from a secure environment. For more details about how to generate and load the key store into device PFR, see [Section 41](#).

7.3.1 PUF key code format

Table 26. PUF key code format

Offset	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, representing a wrapped key. An SRAM PUF device unique key is used as a key wrapping key (KWK).

7.3.2 Key descriptions

Secure binary key encryption key (SBKEK)

- Used for SB2 firmware update image decryption
- 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.

User key encryption key (USERKEK)

- Not used by the MCX W23 bootloader. Available for user as preshared master key
- AES-256 symmetric key
- blhost key type 11
- PUF key index 0

AES key – AES encryption key

- Not used by the MCX W23 bootloader. Available for user as preshared master key
- AES-256 symmetric key
- blhost key type 4
- PUF key index 0

Unique device secret (UDS) key

- 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, the ROM locks PUF index 15.

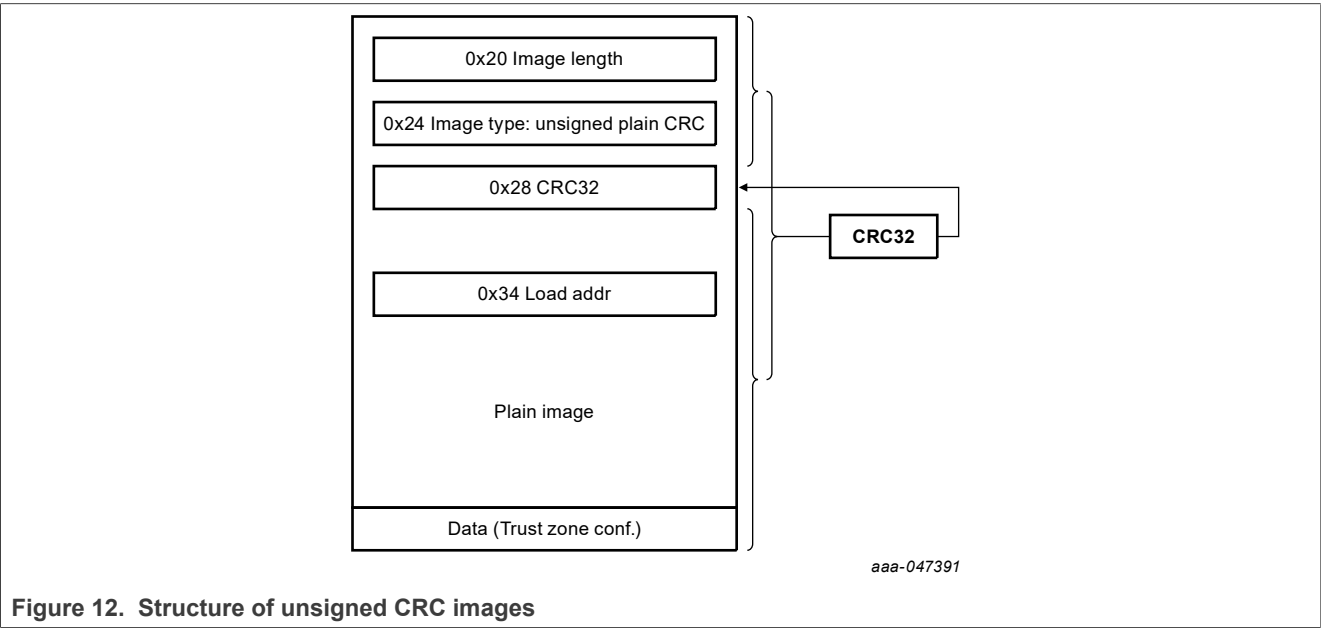
PRINCE region 0-3 key

- 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
- Region 3 - blhost key type 10
- All four keys are PUF key index 0, meaning that the keys unwrap to a dedicated hardware bus, directly connected to the PRINCE cryptographic engine.

7.3.3 Plain image structure

Unsigned plain CRC images are supported by non-secure versions of MCX W23 and secure versions during development life-cycle state of S parts (MCX W23).

Figure 12 shows the structure of unsigned CRC images.

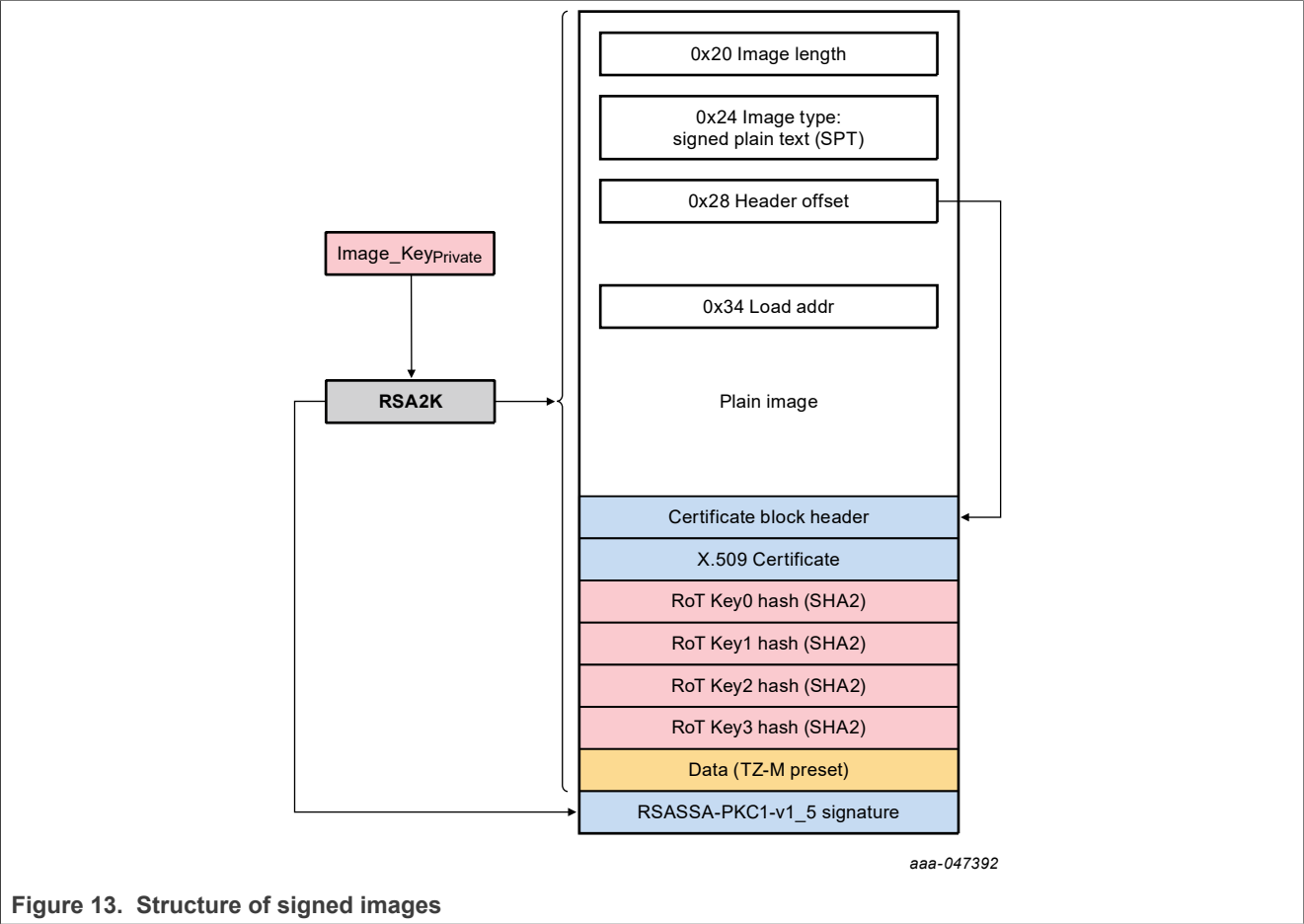


Note: When the image type is 0x0, CRC32 checking is bypassed. Such an image can be used as a generic image during development.

7.3.4 Signed image structure

Images are signed using the RSASSA-PKCS1-v1_5 signature scheme. The digest is computed using SHA-256. 2048-bit or 4096-bit RSA keys are supported.

Figure 13 shows the structure of signed images.



- Image length - total length of the image in bytes including signature
- Image type - signed plain text (SPT) = 0x4 or 0x5

Table 27. MCX W23 image type (word at offset 0x24)

Bit	Name	Value
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 does not use TZ-M.
13	TZ-M preset	0: No TZ-M peripherals preset 1: TZ-M peripherals preset. The bootloader configures TZ-M-related peripherals 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, the word at 0x8028 contains the value 0x1c000.

[Table 28](#) gives a standard Cortex-M33 NVIC vector table with the offset to the certificate block header highlighted.

Table 28. Standard Arm Cortex-M33 NVIC vector table

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	<code>offsetToCertificateBlockInBytes</code>
0x030	SVC
0x034	DebugMon
0x038	reserved
0x03C	SysTick

7.3.5 Certificate block

The certificate block consists of the certificate block header, the certificate table, and the RKH table, concatenated together.

The certificate block can reside anywhere within the signed image. However, it must be fully contained within the signed data, such that the certificate block itself is signed. The most common constructions have the certificate block placed at either the beginning (after the vector table) or at the end of the signed data.

[Figure 14](#) shows the structure of the certificate block.

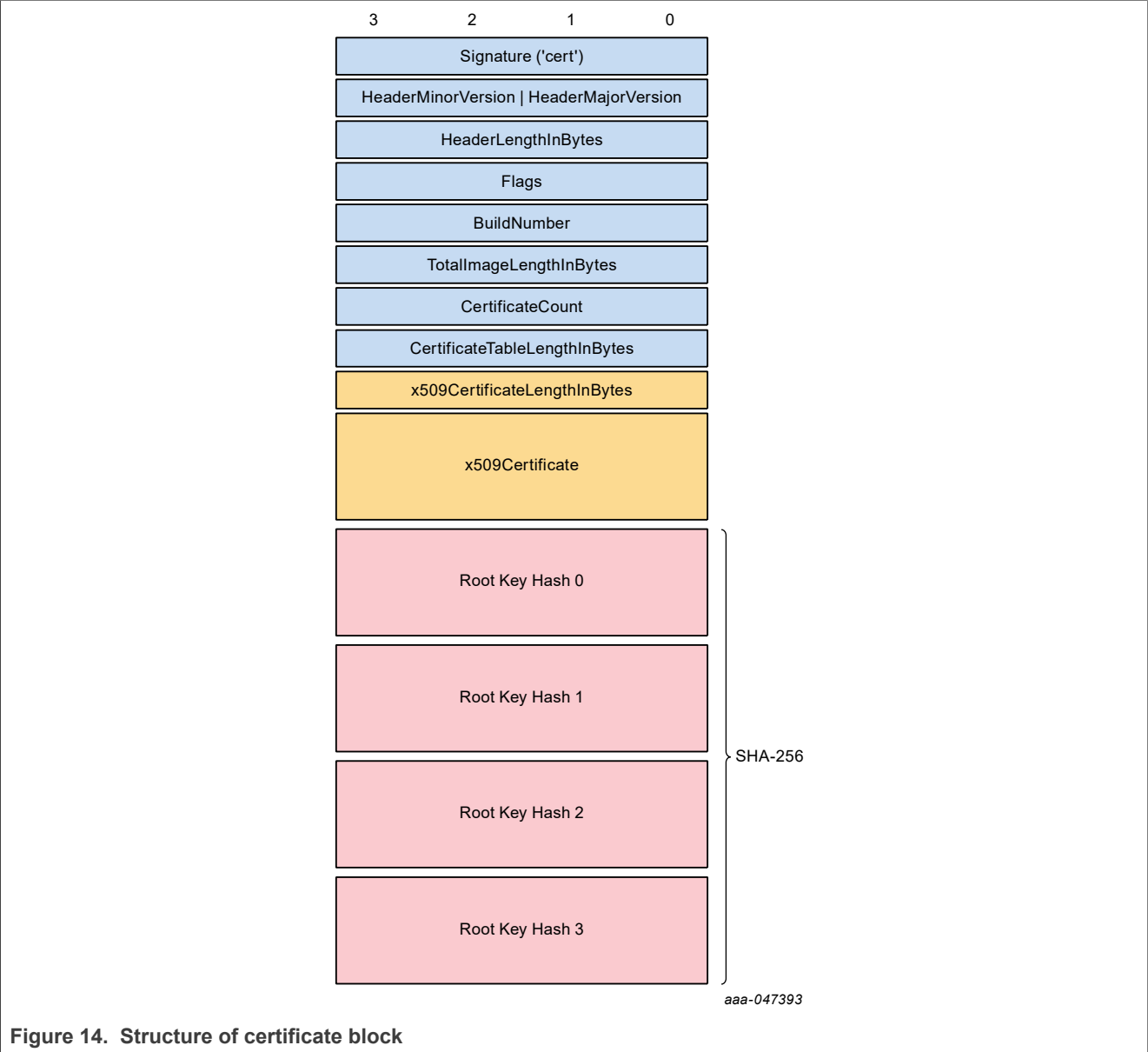


Figure 14. Structure of certificate block

7.3.5.1 Certificate block header

The certificate block header (or just certificate header) is a structure containing information required to verify a signed image properly. As described above, the offsetToCertificateBlockInBytes header offset field points to it. The first word of the certificate block header must be 4-byte aligned.

[Table 29](#) gives descriptions of the fields in the certificate block header.

Table 29. Certificate block header fields

Field	Description
signature	always set to 'cert'
headerMajorVersion	set to 1
headerMinorVersion	set to 0

Table 29. Certificate block header fields...continued

Field	Description
headerLengthInBytes	length of the header in number of bytes, 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 the user to prevent reverting to old versions. The API compares it against the minBuildNumber specified in kb_options_t.
totalImageLengthInBytes	length in bytes of the signed data
certificateCount	must be greater than 0
certificateTableLengthInBytes	total length in bytes of the certificate table

The key field in the certificate header is totalImageLengthInBytes. This field indicates the number of bytes of signed data, starting at offset 0 of the image. The entire certificate block must be contained within the signed data.

The signature field can be treated as a four-character string, without a terminating null byte, with a value of 'cert'. Represented as a 32-bit little-endian constant, the value is $((\text{'c'}) \mid (\text{'e'} \ll 8) \mid (\text{'r'} \ll 16) \mid (\text{'t'} \ll 24))$.

7.3.5.2 Certificate table

Immediately following the certificate block header is the certificate table. It consists of a complete chain of one or more X.509 certificates, each prefixed with a length word.

The offset to the beginning of the certificate table is calculated as `offsetToCertificateTable = (offsetToCertificateBlockInBytes + headerLengthInBytes)`.

The x509CertificateLengthInBytes field for each certificate must be set to the length of the data in bytes of that certificate, rounded up to the next word (4-byte alignment). So, x509CertificateLengthInBytes must be divisible by 4.

x509Certificate contains the actual certificate data and can be of variable length. 0 up to 3 bytes of padding may be 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. It must always be divisible by 4.

There are a number of restrictions on the certificates:

- Only X.509 certificates with v3 format are supported and they must be DER encoded.
- They must use RSA-2048, RSA-3072, or RSA-4096 and SHA-256. RSA-3072 is not recommended for root of trust certificates.
- All certificates must use RSA keys with a modulus bit length greater than or equal the RSA bit length, specified by the security profile. It means that a 4096-bit (or 3072-bit) root of trust 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 of trust 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 are:

- One self-signed, non-CA certificate
- A self-signed root CA certificate, followed by an image signing certificate which the root certificate itself signs.

7.3.6 ROM firmware update using SB file

The secure binary (SB) image format is a command-based firmware update image. It has a long history and has been used on multiple STMP, i.MX, and LPC devices. The MCX W23 supports version 2.1 of the SB image format. 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. It also uses a digital signature mandatory. 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 decryption 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 User's Guide ([Ref. 8](#)). 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.

7.3.6.1 Header

The header contains plaintext information about the SB file, such as version, length, and nonce for AES CTR mode. With SB file version 2.1, the header also contains the RSA signature of the hash digest computed on all the header plain text 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 the ROM can execute during the secure boot flow.

7.3.6.2 MAC of the section MAC table

The SB file contains the expected MAC value computed on the section MAC table. During image verification, the actual MAC of the section MAC table is computed and compared with the expected MAC. The digital signature verification provides the expected MAC value.

7.3.6.3 Key blob

The key blob wraps two 256-bit keys using the RFC3394 algorithm. It provides integrity and authenticity over the wrapped keys. The two keys in the key blob are:

- Data encryption key (K DEK).
- MAC key (K MAC)

The K DEK is used to encrypt section data in the SB file. K MAC is used for header HMAC (if available) and section HMACs.

Both keys are uniquely generated each time an SB file is built. They are wrapped with the SBKEK that is programmed into the PFR of the target device.

7.3.6.4 Sections

The content of SB files is divided into an arbitrary number of sections, each with a unique ID. A boot tag that acts as a header plus an HMAC table precedes every section. A section may either be a bootable section that contains boot commands or a data section containing data the loader does not use. There must be at least one bootable section for an SB file to be valid. The MCX W23 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 that the loader sequentially searches 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 key blob of the SB file. To trade between memory utilization and protection granularity, the user can configure the number of MACs for a section.

Bootable section

A section that has the bootable section flag set is called a bootable section. It contains a sequence of boot commands that the loader processes to perform a firmware update.

The boot commands are described in the elftosb user guide. The MCX W23 ROM loader supports 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 also be used to program internal flash, including the PFR CFPA page, assuming the flash is first erased, for example, using the FlashEraseAll or the FlashEraseRegion commands. The ConfigureMemory command can be used to configure the MCX W23 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

Data section

The loader considers any section with the bootable section flag cleared as data. It does not examine the contents of such sections. It simply skips over them. Optionally, data sections may 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 the 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. So, the image-authenticate function of the ROM 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 SB 2.1 header.

7.3.6.5 Usage of firmware update

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

The recommended method for performing secure firmware updates is:

1. A user application receives an encrypted SB file containing new firmware and stores it in the external SPI flash, or a similar memory.
2. The ROM API is used to authenticate the SB file.
3. The ROM API is then used to decrypt and load the SB file.
4. If secure boot is used as well, the ROM API can be used to authenticate the new firmware in flash before rebooting into it. If this final authentication fails, the new firmware must be made nonexecutable 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 2.1 file processing

The processing of SB 2.1 by ROM depends on the presence of a valid key store setup with the SBKEK key code. Below is an example of SBKEK key store provisioning into the device using the ROM bootloader key provisioning commands.

- Install SBKEK into key store. SBKEK key type = 3.

```
blhost -p COMxx -- key-provisioning set_user_key 3 sbkek.bin
```

- Save the key store into PFR

```
blhost -p COMxx -- key-provisioning write_key_nonvolatile 0
```

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

7.4 ROM TrustZone support

7.4.1 TrustZone image type

From the 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 the image header at offset 0x24, bit 14 (TZM_IMAGE_TYPE):

Table 30. TrustZone image type

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

7.4.1.1 TrustZone disabled image

TrustZone disabled image is an image, which must be executed on devices without TrustZone (CM33 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 MCX W23 ROM supports direct execution of software developed for CM33 devices without security extension. If the TrustZone disabled image is executed, the ROM configures all device resources into normal world, locks access to all TrustZone related configuration registers, switches from secure to normal world, and finally jumps to user application before it jumps to the user application. This mode allows easy reuse of the software developed for Cortex-M33 without security extension. The user does not have to perform any software modification.

Note: After the jump into the user application, the security extension (TrustZone) is still enabled. The MCU runs in normal mode, all TrustZone related configuration registers are locked, and memory region 0x13004000-0x13005000 (first 4 kB of ROM) is configured as secure memory. Code execution or data read from this memory must be avoided. Otherwise, a HardFault is generated.

7.4.1.2 TrustZone enabled image

A TrustZone enabled image is an image that is executed on devices with TrustZone (M33 with security extension) and that uses TrustZone. In this case, the user application is split into secure and non-secure partitions. After a device reset, the software execution starts in secure mode. If the TrustZone enabled image is executed, the ROM does not set any TrustZone settings (except optional TrustZone preset data configuration as described in the next section) and jumps into the user application in secure mode. The executed software/image is responsible for TrustZone settings and jump from secure to normal mode.

7.4.2 TrustZone preset data

The MCX W23 ROM supports TrustZone data configuration during the boot process. The TrustZone preset data includes:

- VTOR, VTOR_NS, NVIC_ITNS0, NVIC_ITNS1 registers
- Secure MPU
- Non-secure MPU
- SAU
- Secure AHB controller

If the TrustZone preset is enabled, the ROM applies the presets after image validation. The ROM configures all TrustZone-related registers using the preset data provided at the end of the image. If any register or whole peripheral has a lock bit and the corresponding bit is set in the preset data, the register or whole peripheral 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 preconfigured TrustZone environment and it does not have to contain any TrustZone configuration code.

7.4.2.1 TrustZone preset data structure

The C-structure below defines the TrustZone preset data 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_rule0;          /*!< RAM2 Memory Rule Register 0 */
uint32_t ram2_mem_rule1;          /*!< RAM2 Memory Rule Register 1 */
uint32_t spifi_slave_rule;        /*!< SPIFI Slave Rule Register */
uint32_t spifi_mem_rule0;         /*!< SPIFI Memory Rule Register 0 */
uint32_t spifi_mem_rule1;         /*!< SPIFI Memory Rule Register 1 */
uint32_t spifi_mem_rule2;         /*!< SPIFI Memory Rule Register 2 */
uint32_t spifi_mem_rule3;         /*!< SPIFI Memory Rule Register 3 */
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_port7_slave_rule0;     /*!< AHB port 7 Slave Rule Register 0 */
uint32_t ahb_port7_slave_rule1;     /*!< AHB port 7 Slave Rule Register 1 */
uint32_t ahb_port8_slave_rule0;     /*!< AHB port 8 Slave Rule Register 0 */
uint32_t ahb_port8_slave_rule1;     /*!< AHB port 8 Slave Rule Register 1 */
uint32_t ahb_port9_slave_rule0;     /*!< AHB port 9 Slave Rule Register 0 */
uint32_t ahb_port9_slave_rule1;     /*!< AHB port 9 Slave Rule Register 1 */
uint32_t ahb_port10_slave_rule0;    /*!< AHB port 10 Slave Rule Register 0 */
uint32_t ahb_mem_rule0;             /*!< AHB Memory Rule Register 0 */
uint32_t sec_gp_reg0;               /*!< Secure GPIO Register 0 */
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`. [Table 31](#) gives the definition of the configuration word `misc_tzm_settings`.

Table 31. Miscellaneous TZM settings

Bit	Name	Function
31-1		reserved
0	SECUREFAULTENA	copied into the Arm Cortex M33 SHCSR.SECUREFAULTENA control bit 0b - Secure faults disabled 1b - Secure faults enabled

The configuration data are attached in binary format at the end of the image, if a signed image is located in front of the signature (see [Figure 15](#)).

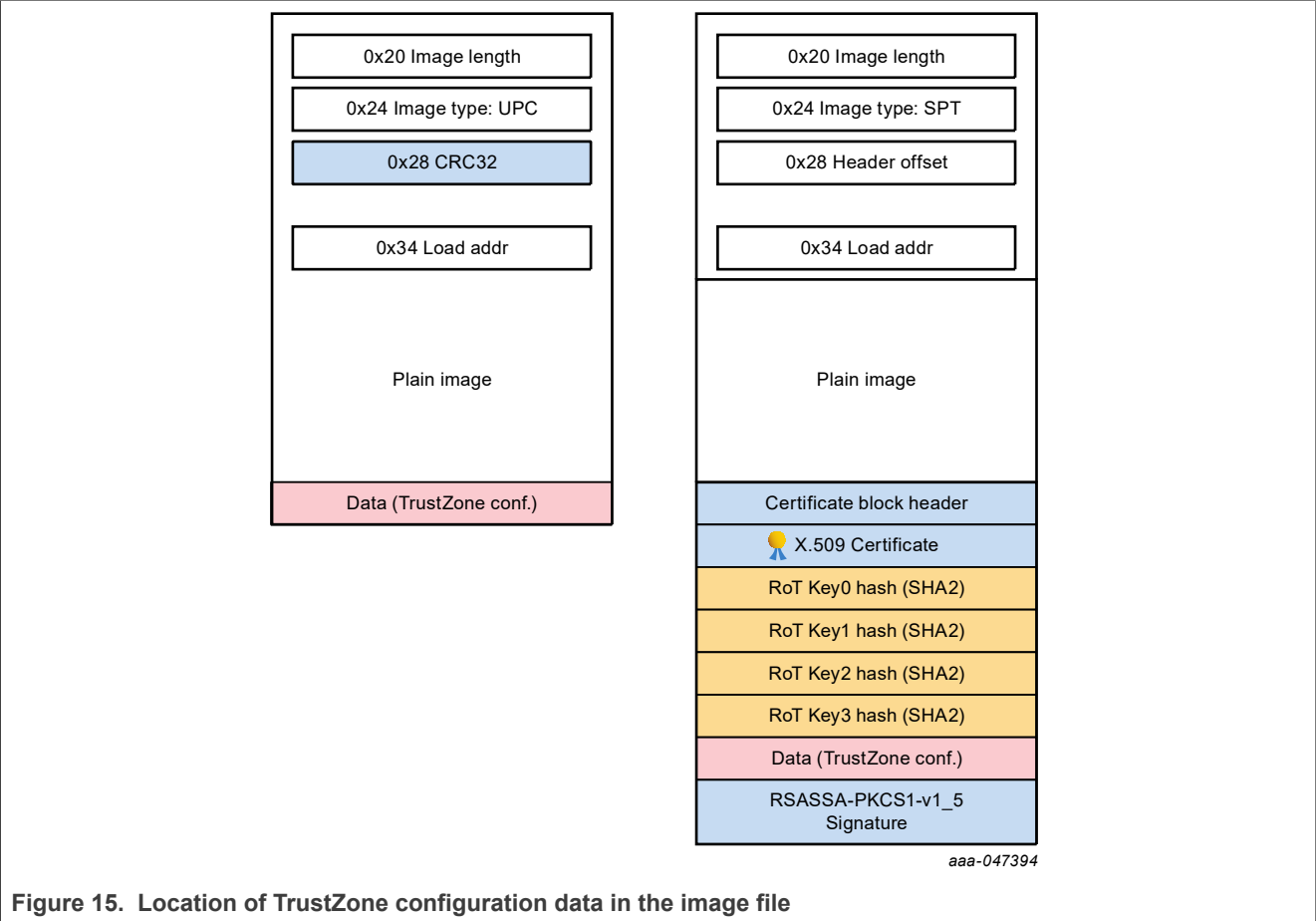


Figure 15. Location of TrustZone configuration data in the image file

The elftosb tool can also be used to create TrustZone configuration data. For more information, see the elftosb user guide.

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

Table 32. TZM Preset value

TZM_PRESET value (offset 24, bit 13)	
0	TrustZone data not present
1	TrustZone data present

Note: The TrustZone configuration is enabled before a jump to the user application. So, the TrustZone configuration data of the user must allow ROM code execution for successful transition from secure to normal mode and jump to user application. The TrustZone settings of the user must include the following:

- The whole ROM space (0x13000000-0x1301FFFF) must be configured as secure privilege.
- 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.

7.4.2.2 TrustZone image type restriction control during boot process

The user can restrict, which TrustZone image type is allowed for execution. For this purpose, there are two bits TSM_IMAGE_TYPE in the SECURE_BOOT_CFG field of the CMPA. These bits restrict the execution of the image depending on its TrustZone image type. [Table 33](#) describes this process.

Table 33. Allowed TrustZone image types

Allowed TrustZone image type	TSM_IMAGE_TYPE value
Any TrustZone image type is allowed. The image type is set in the vectors section of the image (offset 0x24)	00b
Only images with TrustZone disabled are allowed.	01b
Only images with TrustZone enabled are allowed, with or without TrustZone preset data.	10b
Only images with TrustZone enabled with preset data are allowed	11b

7.4.3 Boot ROM API and TrustZone

7.4.3.1 TrustZone disabled images

A TrustZone disabled image is executed in normal mode. The whole memory space is configured as non-secure (except the first 4 kB of ROM). So, the ROM API can be used without any limitation.

7.4.3.2 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. It means that code executed in normal world can program data into flash memory, but it cannot erase the whole flash or reprogram the application itself or its secure part.

To enable the ROM API into normal world, the user must create an entry function for every ROM API function exposed to normal world. An 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[4])
__cmse_nonsecure_entry status_t flash_program_NSE(flash_config_t
*config, uint32_t start,
uint32_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 the user can call the flash program function from the normal world as:

```
flash_config_t flashConfig;  
uint32_t programBuffer[144];  
status = flash_program_NSE(&flashConfig, 0x0, programBuffer,  
    sizeof(programBuffer));
```

7.5 Secure boot usage

The MCX W23 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 the level of protection against attacks.

The ROM also supports public keys and image revocation, that is, the method of not allowing new updates to be applied unless they are of a specific version, which is the basis for rollback protection.

The following section describes the main steps for key provisioning, creating signed images and loading the signed images into the target. Tools used are elftosb or elftosb-gui. For a step-by-step guide describing the use of these tools, see the LPC55Sxx Secure Boot application note ([Ref. 6](#)).

7.5.1 Keys and certificates

The image signing process requires an RSA key pair and an image signing certificate. For example, use openssl for key and certificate generation.

ROM supports:

- Up to 4 root of trust (RoT) keys with RoT key revocation feature
- Up to 16 image key certificates with image revocation feature

Before the secure image preparation, the root key hash table must be written to corresponding CFPA boot pages.

7.5.1.1 CFPA/CMPA page preparation

Before using of the device for the first time, the CFPA and CMPA pages are cleared. Also, registers related to secure boot must be set up.

The RKTH_REVOKE field at CFPA page address 0xFE018 must be set up to accept signed images with created certificates.

1. Enter ISP boot mode by asserting ISP boot pin.
2. Prepare CMPA page using elftosb-gui PC tool.
 - RKTH field containing the root key table hash
 - SEC_BOOT_EN secure boot enable bits
 - RSA4K bits selecting minimal key length
3. Prepare CFPA page using elftosb-gui PC tool.
 - RKTH_REVOKE field to accept signed images with created certificates
4. Write prepared CFPA/CMPA page into flash memory using blhost tool.

7.5.1.2 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 step-by-step guide describing the use of these tools, see the LPC55Sxx Secure Boot application note ([Ref. 6](#)).

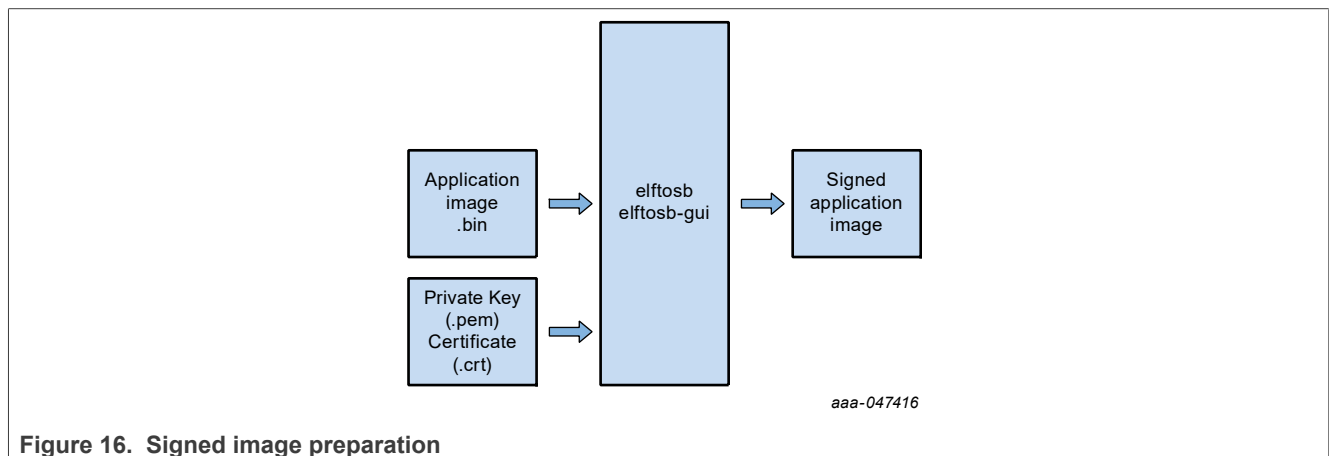


Figure 16. Signed image preparation

To produce an internal flash (XIP) signed image, the elftosb tool required the following information:

- Plain application binary generated for the MCX W23 device
- Start address of the application binary
- TrustZone related settings
- Root of trust certificates, and optionally chain certificates
- Private key for selected certificate (last certificate in chain)

7.5.1.3 Loading signed image

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

- ROM in-system programming (ISP) using the write-memory blhost command
- ROM ISP using the secure FW update container (SB2 file)
- Programming signed image directly from target application using ROM API
- Flashing signed image through debugger

7.5.2 Internal flash encryption using PRINCE engine

The boot ROM offers configuration of the PRINCE engine during boot time.

First, during the flash programming, the PRINCE engine is set up to store the image in an encrypted format using a user-defined key.

The key is stored on the device and protected against copying using PUF encryption, in a format only readable for a given instance of the processor. SRAM PUF internally uses HW-specific random keys unique for each device.

During boot, ROM locates the key store, decrypts it internally using SRAM PUF, and preconfigures 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 steps describe the key provisioning and how to program the application image using PRINCE.

7.5.2.1 PRINCE-related PUF key store setup

The keys used for PRINCE encryption/decryption are generated in the device using on-chip SRAM PUF. They are delivered to the PRINCE engine through the internal hardware bus.

The MCX W23 contains four preconfigured, unique PRINCE keys in the NXP area of the PFR. The boot ROM automatically installs them for use by the application.

7.5.2.2 PRINCE region configuration with blhost

For PRINCE encryption and decryption, the regions and subregions for the crypto operation must be configured. The configuration is accomplished with the ISP "configure-memory" command or by using the PRINCE ROM API (see [Section 7.6.2](#)). This command must be called with the data structure and ancillary PRINCE commands shown in the following tables and according to the following procedure.

Table 34. Structure for configure-memory command

Offset	Size	Description
0	4	PRINCE configuration
4	8	PRINCE region info

Table 35. PRINCE configuration register for configure-memory command

Bit	Symbol
1:0	0x00 – PRINCE region 0 0x01 – PRINCE region 1 0x10 – PRINCE region 2 0x11 – PRINCE region 3
25:2	reserved
31:8	0x50 ('P') – configure PRINCE

Table 36. PRINCE region info register for configure-memory command

Bit	Symbol
31:0	PRINCE region X start
63:32	PRINCE region X size

Load structure into RAM memory and call "configure-memory" command with this structure:

1. Region selection (region 0 in this example)
 - `blhost.exe -p COMxx -- fill-memory 0x2000C000 4 0x50000000`
2. Start address of encrypted area (address 0x0 in this example)
 - `blhost.exe -p COMxx -- fill-memory 0x2000C004 4 0`
3. Length of the encrypted area (0x40000 in this example)
 - `blhost.exe -p COMxx -- fill-memory 0x2000C008 4 0x40000`
4. Call configure-memory with prepared structure in RAM
 - `blhost.exe -p COMxx -- configure-memory 0 0x2000C000`

Following this command, PRINCE is configured for flash encryption.

Note: The PFR area must be excluded from PRINCE encryption area, that is, the start and size settings in the configuration for the structure must be set to avoid overlapping with the PFR area.

7.5.2.3 Upload image

A "PRINCE erase checker" is implemented in the boot ROM that checks if all PRINCE subregions (8 kB blocks) are erased at once. Similarly, a "PRINCE flash write checker" is implemented in the ROM code to check if all PRINCE subregions (8 kB blocks) 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 internal 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`
4. The image loaded into internal flash starts executing.

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 must encompass the whole encrypted area previously defined through the configure-memory command. In other words, the image (.bin) size must be equal to the encrypted area size (8 kB aligned). Each entire PRINCE encrypted flash area must be erased and written in a single operation. It is recommended that for performing PRINCE region configuration as well as erase and write operations the latest MCUXpressoSDK PRINCE driver is used.

7.6 Secure ROM API

The ROM API table is located at address 0x1301FE00 and contains pointers to the several exposed ROM APIs. The ROM APIs contain absolute ROM API function addresses, which can be called using function pointers. The PRINCE ROM API section starts at address 0x1300507C. Only secure boot related functions are described in this chapter. The flash-related ROM API functions are described in [Section 9](#).

The main purpose of these APIs is to provide access to functions used and implemented in ROM to authenticate the application image, to configure the PRINCE on-the-fly encryption/decryption peripheral, and to process SB files.

Table 37. Secure ROM API summary

Address in ROM API table	Absolute function address	Function
0x130050E4	0x1300B663	skboot_status_t kb_authenticate(const uint8_t *imageStartAddr, secure_bool_t *isSignVerified)
0x130050E8	0x13006591	void hashcrypt_irq_handler(void)
0x1300507C	0x1300989B	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)
0x130050D4	0x13012F41	status_t kb_init(kb_session_ref_t **session, const kb_options_t *options)
0x130050D8	0x130130DD	status_t kb_deinit(kb_session_ref_t **session)
0x130050DC	0x13013105	status_t kb_execute(kb_session_ref_t **session, const uint8_t *data, uint32_t data Length)

7.6.1 Image authentication API

7.6.1.1 kb_authenticate

The `kb_authenticate` 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. It also uses it to verify authenticity of the SB 2.1 files. If a user application calls `kb_authenticate()` directly or indirectly from SB file processing functions `kb_init/kb_execute/kb_deinit`, the user HASH interrupt vector calls the `hashcrypt_irq_handler()` function for handling of the hash-crypt IP interrupt. The reason is that the hashing is implemented as non-blocking to achieve shorter computation time. While the hash-crypt AHB master fetches data for hashing, the CPU and the CASPER coprocessor work on RSA verification.

It is important to note that the `skboot_authenticate()` ROM function uses global variables in RAM.

So, the caller must ensure that it does not have any data in the global variables location before the function call. The caller discards the data in the global variables location after the function returns.

The RAM reserved space for ROM global variables on this MCX W23 product is: 0x30004000 to 0x30007FFF

The function requires the `imageStartAddr` input pointer to be 32-bit word aligned. The status is returned in two ways, via a function return and by a write to the `*isSignVerified` pointer. It offers redundant protection. The caller verifies both return values and only considers the image to be authentic when the function returns `kStatus_SKBOOT_Success` AND `*isSignVerified == kSECURE_TRACKER_VERIFIED`.

The status code is one of following:

Table 38. kb_authenticate status codes

Status code	Value	Description
kStatus_SKBOOT_Success	0x5ac3c35a	The command completed successfully.
kStatus_SKBOOT_Fail	0xc35ac35a	Parsing certificate header, certificate/certificates chain, RKH, or signature verification failed.
kStatus_SKBOOT_InvalidArgument	0xc35a5ac3	The image contains an unexpected value.

On function output, it writes:

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

*isSignVerified = kSECURE_TRACKER_VERIFIED (0x55aacc33U) when signature verification passes.

7.6.1.2 hashcrypt_irq_handler

The `hashcrypt_irq_handler` function must be called from the user hash-crypt interrupt handler, if the `kb_authenticate()` or `kb_execute()` ROM function is called in user applications. The hashing in ROM is implemented as non-blocking. So, the interrupt handler function is required to assist with fetching data for hash-crypt hardware module.

7.6.2 PRINCE ROM API

The boot ROM API supports the configuration of PRINCE encryption regions. This section describes all PRINCE-related functions that can be called from the user application.

The whole ROM API table is located at address 0x1301fe00 and the PRINCE ROM API part is located at address 0x1300507C.

The bus crypto engine (PRINCE) ROM API prototypes are:

```
typedef struct BusCryptoEngineInterface
{
    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;
```

7.6.2.1 bus_crypto_engine_set_encrypt_for_address_range

The bus_crypto_engine_set_encrypt_for_address_range API function allows the encryption/ decryption for a specified address range. It configures the PRINCE registers and related PFR regions. Before calling this PRINCE API, the flash_init ROM API function must be called. The PRINCE configuration can be also done using the blhost ISP command interface (see [Section 7.5.2](#)).

```
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 39. Parameters

Parameter	Description
region_number	Bus encryption engine region index (0, 1, 2, 3).
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.

The status code is one of following:

Table 40. bus_crypto_engine_set_encrypt_for_address_range status codes

Status code	Value	Description
kStatus_SKBOOT_Success	0x5ac3c35a	The command completed successfully.
kStatus_SKBOOT_Fail	0xc35ac35a	An error occurred.

7.6.3 SB file processing API

7.6.3.1 kb_init

The `kb_init` function initializes the SB file processing API, based on the options the applications provides in the second argument. Every call to `kb_init()` must be paired with a call to `kb_deinit()`.

Prototype

```
status_t kb_init(kb_session_ref_t **session, const kb_options_t *options);
```

Table 41. Parameters

Parameter	Description
session	The function fills in the session pointer with a context that is used for subsequent calls to <code>kb_execute()</code> and <code>kb_deinit()</code> .
options	The options control the operation <code>kb_execute()</code> performs.

The options contain a loadSB structure that contains the following:

- The `buffer` field is a pointer to memory provided by the caller for use during execution of the operation. The minimum size is the size of each certificate in the chain, plus another 432 bytes per certificate.
- The `profile` field is a mask that specifies which features are required in the SB file or image that is processed. It includes the minimum AES and RSA key sizes. See the `_kb_security_profile` enum for profile mask constants. The image loaded or authenticated must match the profile or an error is returned.
- The `minBuildNumber` is an optional field that can be used to prevent version rollback. The API checks the build number of the image. If it is less than `minBuildNumber`, the API check fails with an error.
- The `regions` field points to an array of memory regions that the SB file that is loaded is allowed to access. If the field `regions` is NULL, the SB file can access all memory. This feature is required to prevent a malicious image from erasing good code or RAM contents while it is loaded. And before it is detected that the image is inauthentic when we hit the end of the section.
- The `userSBKEK` field lets users provide their own AES-256 key for unwrapping keys in an SB file during the `kKbootLoadSB` operation. `userSBKEK` must point to a 32-byte AES-256 key. If `userSBKEK` is NULL, the PFR SBKEK is used. After `kb_init()` returns, the caller must zero out the data pointed to by `userSBKEK`, as the API has installed the key in the CAU3.

The status code is one of following:

Table 42. kb_init status codes

Status code	Value	Description
kStatus_Success	0	The command completed successfully.
kStatus_InvalidArgument	4	One of the provided arguments is invalid.

Table 42. kb_init status codes...continued

Status code	Value	Description
kStatus_RomApiInvalidBuffer	10803	The provided buffer is invalid (for example, not properly aligned).
kStatus_RomApiBufferSizeNotEnough	10802	The provided buffer is too small.
kStatus_SKBOOT_KeyStoreMarkerInvalid	0xc3c35a5a	Provisioning of the device has not been done.

7.6.3.2 kb_deinit

The `kb_deinit` function finalizes the operation and cleans up the context.

Prototype

```
status_t kb_deinit(kb_session_ref_t *session);
```

Table 43. Parameters

Parameter	Description
session	The session pointer filled in by a previous call to <code>kb_init()</code> .

The status code is one of following:

Table 44. kb_deinit status codes

Status code	Value	Description
kStatus_Success	0	The command completed successfully.
kStatusRomLdrJumpReturned	10110	The image started by the SB file has returned.
kStatusRomLdrUnexpectedCommand	10117	The SB file contains an unexpected or unsupported command.
kStatusMemoryCumulativeWrite	10203	Writing to flash failed because the flash is not in erased state.
kStatus_FLASH_CompareError	122	The flash program verification failed (only done when the property <code>verifywrites</code> is set to 1).

7.6.3.3 kb_execute

The `kb_execute` function performs the operation configured during `kb_init`. The application must call this function repeatedly, passing in sequential chunks of data from the boot image (SB file) that must be processed. The ROM performs the selected operation on this data and return. The application may call this function with as much or as little data as it wishes. This amount of data can be used to select the granularity of time given to the application in between executing the operation.

Prototype

```
status_t kb_execute(kb_session_ref_t *session, const uint8_t *data, uint32_t dataLength);
```

Table 45. Parameters

Parameter	Description
session	The session pointer filled in by a previous call to <code>kb_init()</code> .
data	Buffer of SB file data provided to the ROM by the application.
dataLength	Length in bytes of the data in the buffer provided to the ROM.

The status code is one of following:

Table 46. `kb_execute` status codes

Status code	Value	Description
kStatus_Success	0	The command completed successfully.
kStatus_Fail	1	An unspecified error occurred while processing the SB file.
kStatus_RomApiExecuteCompleted	0	The ROM successfully processes the whole SB file.
kStatus_RomApiNeedMoreData	10801	No error occurred, but the ROM requires more data to continue processing the SB file.
kStatusRomLdrSignature	10101	The signature or the SB file version is incorrect.
kStatusRomLdrSectionLength	10102	The section chunk count is incorrect.
kStatusRomLdrRollbackBlocked	10115	The user build number, non-secure firmware version, or secure firmware version is smaller than expected.
kStatusRomLdrEOFReached	10104	The SB file size is smaller than expected.
kStatusRomLdrUnexpectedCommand	10117	The SB file contains an unexpected or unsupported command.
kStatusRomLdrUnknownCommand	10107	The SB file contains an unknown command.
kStatusRomLdrChecksum	10105	The SB file contains a command with an incorrect checksum.
kStatusRomLdrInvalidSectionMacCount	10116	The section MAC count is incorrect.
kStatusRomLdrIdNotFound	10108	The SB file does not contains a valid boot section.
kStatusRomLdrCrc32Error	10106	The SB file contains an incorrect checksum.

8 In-system programming (ISP) and In-application programming (IAP)

All MCX W23 devices include in-system programming (ISP) functions to support image programming from serial interface (UART, I²C, SPI). In-application programming (IAP) functions are available in ROM to user applications running on the device.

8.1 Features

In-system programming supports:

- UART, I²C, and SPI peripheral interfaces
- Automatic detection of the active peripheral
- Automatic baud rate detection for the UART peripheral
- 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
- Commands 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
- Commands for internal flash
- Commands for encrypted image download

8.2 General description

8.2.1 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 low-power mode (PowerOff, DeepPowerDown, or PowerDown).

The bootloader provides a flash programming utility that operates over a serial connection on the MCU. It enables quick and easy programming of the MCU through the entire product lifecycle, including application development, final product manufacturing, and more. To communicate with the bootloader, host-side command lines and GUI tools are available. Users can use host tools to upload/download application code and do manufacturing via the bootloader.

For the bootloader operation and boot pin (PIO0_20/ISP), see [Section 6](#).

8.2.2 In-system programming (ISP) and in-application programming (IAP)

In-system programming is supported in several different ways:

- For details of the ISP protocol, see [Section 8.3](#).
- For details of the ISP packet, see [Section 8.4](#).
- For details of the ISP commands, see [Section 8.5](#).
- For details of UART in-system programming, see [Section 8.6](#).
- For details of I²C in-system programming, see [Section 8.7](#).
- For details of SPI in-system programming, see [Section 8.8](#).

In-application programming is done using the ROM API.

- The flash-related ROM API functions are described in [Section 9](#).
- The ROM API functions related to security are described in [Section 7.6](#).

8.2.3 Memory map after any reset

The boot ROM is located in the memory region starting from the address 0x1300 0000. The ISP software and the IAP software use parts of the on-chip RAM. The RAM usage is described in [Section 8.2.4](#). For more information, see [Section 6](#).

Based on the DEFAULT_ISP_MODE bit settings or ISP pin settings, the ROM enters the ISP mode and autodetect activity on the I²C, SPI, or USART interface. The autodetect looks for activity on the USART, I²C, and SPI interfaces. When a properly formed frame is received, it selects the appropriate interface. If an invalid frame is received, the data is discarded and scanning resumes. USART, I²C, and SPI ISP communications are described in [Section 8.3](#) and [Section 8.4](#).

8.2.4 Interrupts and SRAM usage

8.2.4.1 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.

The IAP calls can take a considerable time to return. So, using the IAP calls must not be done at the same time as other time-critical operations.

8.2.4.2 RAM usage

[Figure 17](#) shows the regions that are reserved for bootloader use when the bootloader is running. Before calling the ROM APIs in user application (typically by marking this RAM as reserved in the application linker file), the heap and the BSS, RW section must be reserved for the ROM API use.

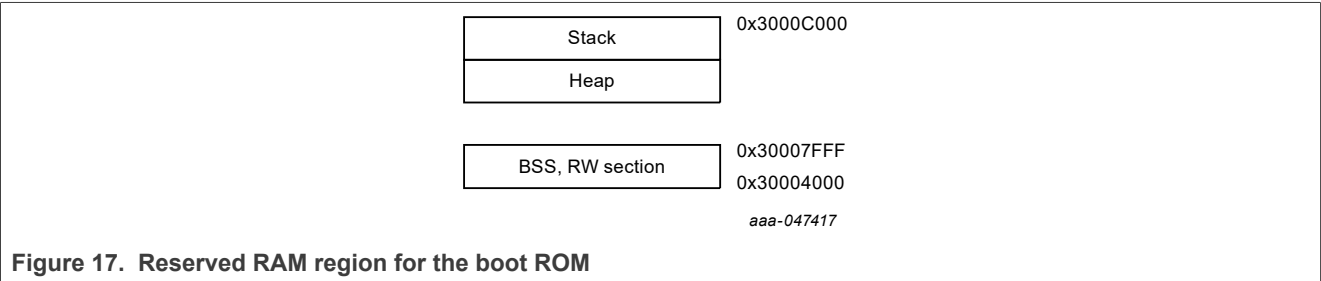


Figure 17. Reserved RAM region for the boot ROM

8.3 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. Each command or response packet must be acknowledged with an ACK (framing packet with packet type set to kFramingPacketType_Ack, see [Section 8.4.2](#) for details).

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.

8.3.1 Command with no data phase

The protocol for a command with no data phase contains (ACK framing packets not mentioned):

- Command packet (from the host)
- Generic response command packet (to host)

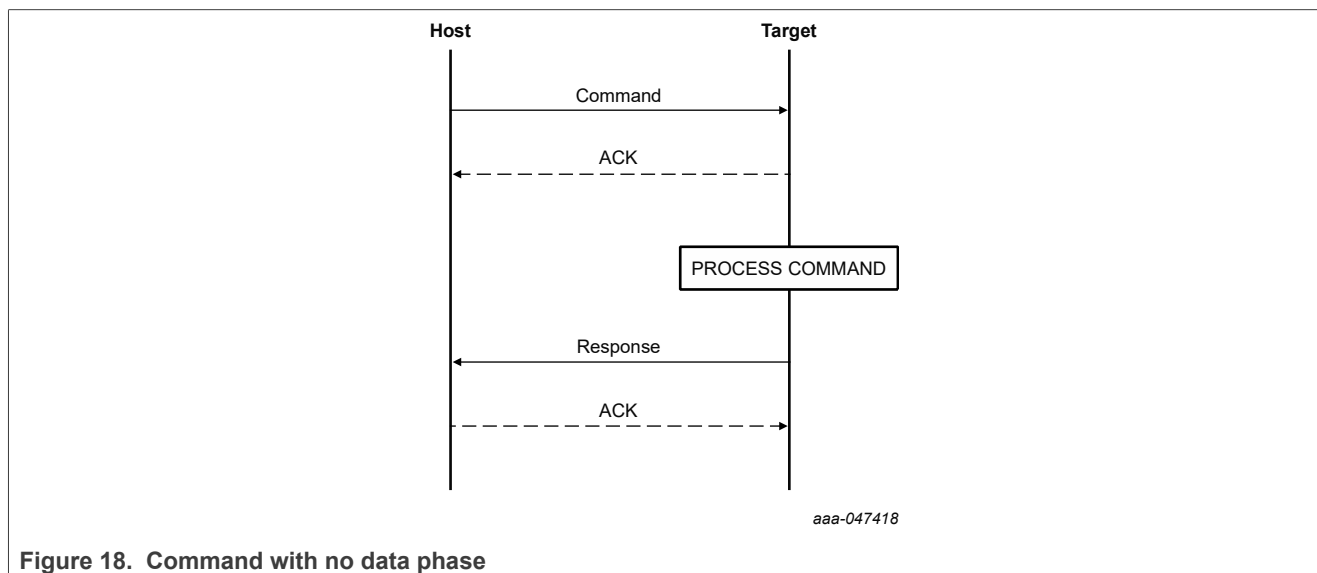


Figure 18. Command with no data phase

Note: 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 been processed.

8.3.2 Command with incoming data phase

The protocol for a command with incoming data phase contains (ACK framing packets not mentioned):

- Command packet (from host) (kCommandFlag_HasDataPhase set, see [Section 8.4.6](#) for details)
- Generic response command packet (to host)
- Incoming data packets (from the host)
- Generic response command packet (to host)

Notes

- The host may not send any further packets while it is waiting for the response to a command.
- If the generic response packet prior to the start of the data phase does not have a status of kStatus_Success, the data phase is aborted.
- The receiving side may abort data phases 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.

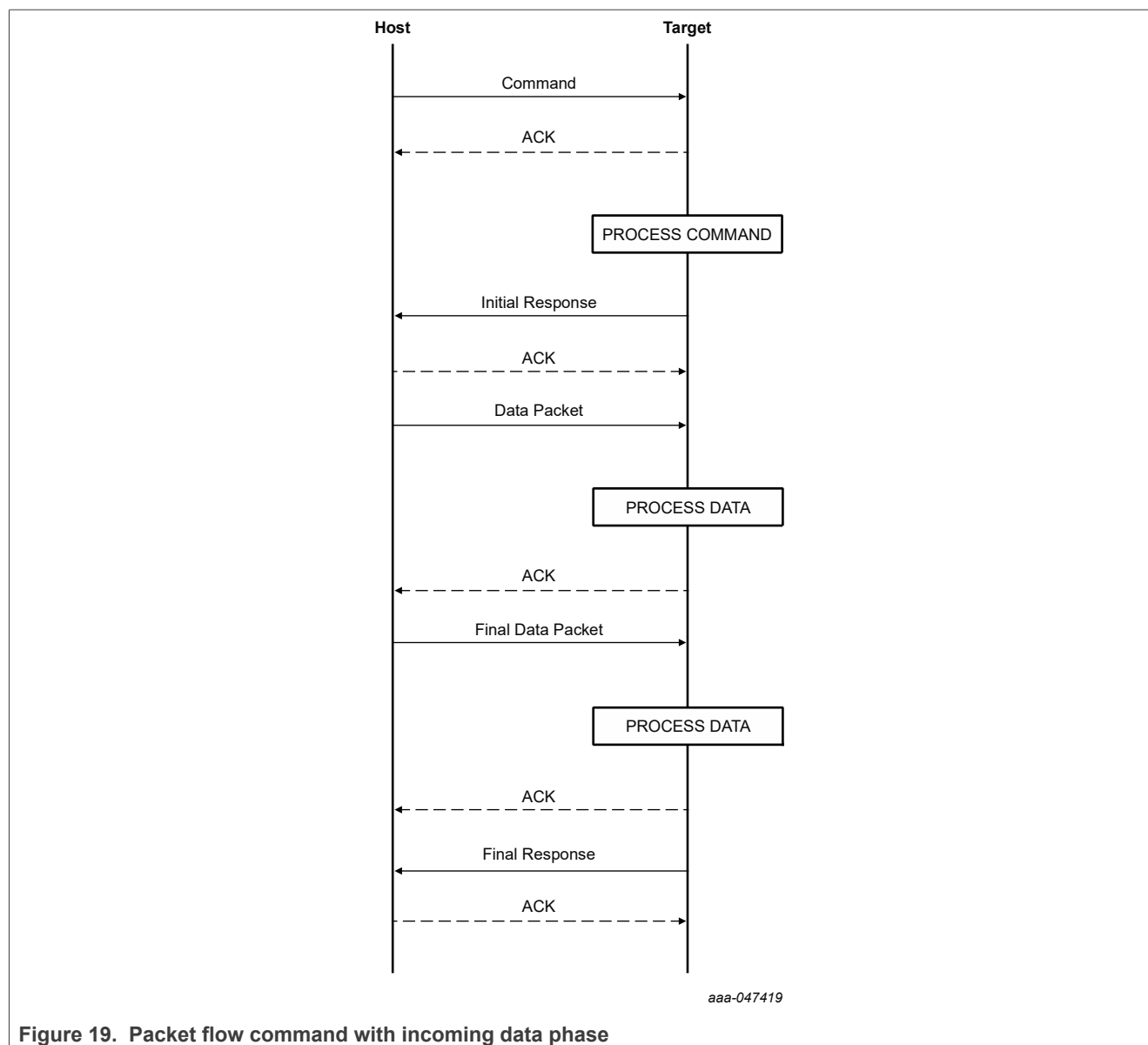


Figure 19. Packet flow command with incoming data phase

8.3.3 Command with outgoing data phase

The protocol for a command with an outgoing data phase contains (ACK framing packets not mentioned):

- Command packet (from the host)
- Response command packet (to host) (kCommandFlag_HasDataPhase set, see [Section 8.4.6](#) for details)
- Outgoing data packets (to host)
- Generic response command packet (to host)

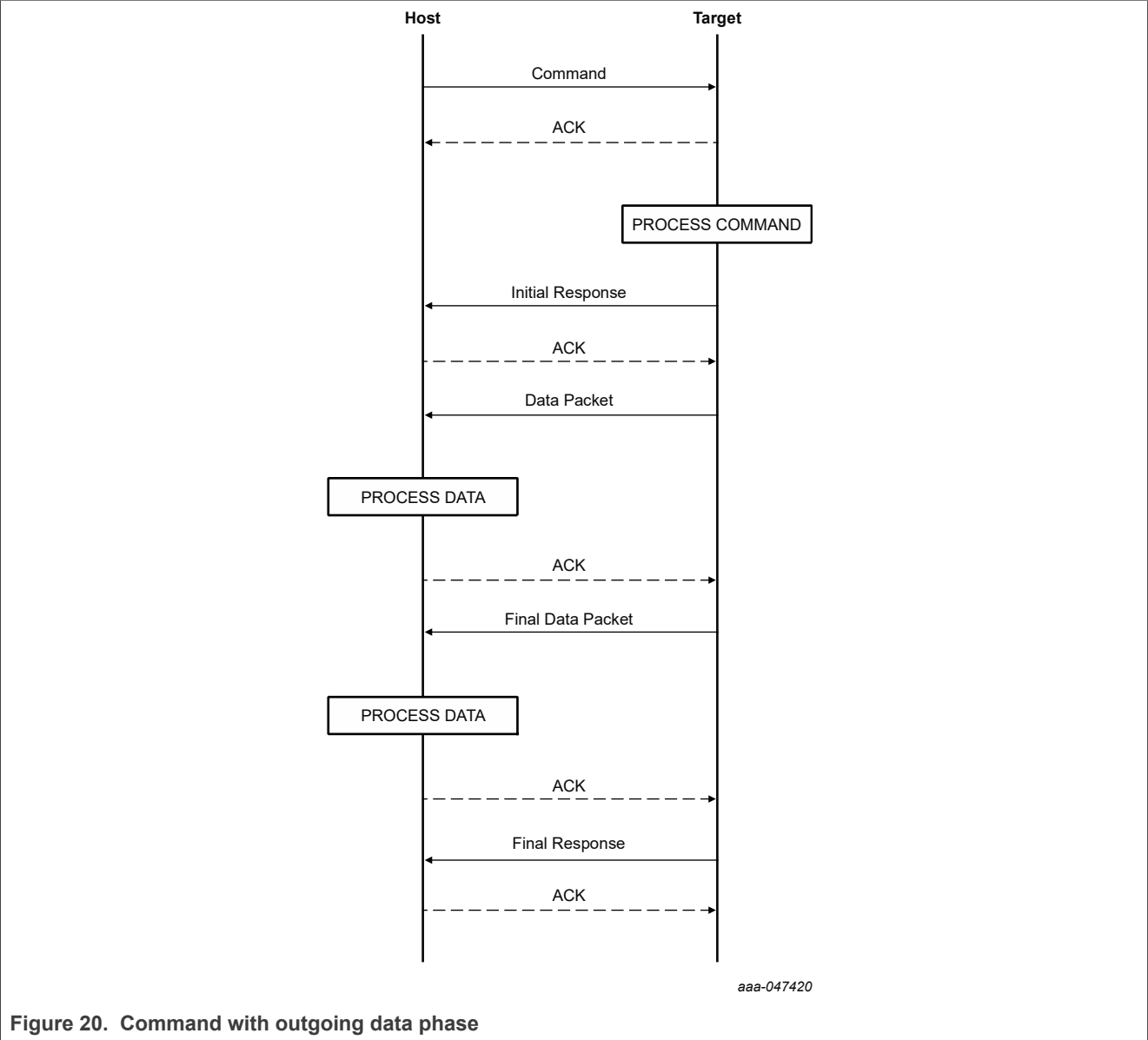


Figure 20. Command with outgoing data phase

Notes

- 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.
- If the response command packet does not contain the kCommandFlag_HasDataPhase flag before the start of the data phase, the data phase is aborted.
- The host may abort data phases by 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.

8.4 Bootloader packet types

8.4.1 Introduction

The bootloader device works in slave mode. A host, either a PC or an embedded host, initiates all data communications. The bootloader device is the target, which receives a command or data packet. All data communications between host and target are packetized and embedded in a framing packet.

There are different types of packets used:

- Framing packet, see [Section 8.4.2](#)
- Ping packet, see [Section 8.4.4](#)
- Ping Response packet, see [Section 8.4.5](#)
- Command packet, see [Section 8.4.6](#)
- Response packet, see [Section 8.4.7](#)
- Data packet, see [Section 8.4.8](#)
- ACK packet, see [Section 8.4.9](#)

All fields in the packets are in little-endian byte order.

8.4.2 Framing packet

The framing packet is used for flow control and error detection for the communications links that do not have such built-in features. 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, I²C, and SPI.

Table 47. Framing packet format

Byte	Value	Parameter	Description
0	0x5A	Start byte	
1	-	PacketType	see Table 49 for possible values
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	16-bit field The CRC16 value covers entire framing packet, including the start byte and command or data packets. However, it does not include the CRC bytes (see Section 8.4.3).
5	-	Crc16_high	

Table 47. Framing packet format...continued

Byte	Value	Parameter	Description
6...n	-	command or data packet payload	

A special framing packet that only contains a start byte and a packet type is used for synchronization between the host and target (see [Section 8.4.4](#) and [Section 8.4.9](#)).

Table 48. 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 49. 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 resent.
0xA3	kFramingPacketType_AckAbort	Data phase is aborted.
0xA4	kFramingPacketType_Command	The framing packet contains a command packet payload. The command packet is also used for responses.
0xA5	kFramingPacketType_Data	The framing packet contains a data packet payload.
0xA6	kFramingPacketType_Ping	Sent to verify that 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.

An example of a wrapped command packet (for the [Section 8.5.2](#)) shows how a framing packet is used. The parameters in green are included in the CRC calculation.

Table 50. GetProperty command packet format (example)

GetProperty	Parameter	Value
framing packet	start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	Crc16	0x4B 0x33
command packet	commandTag	0x07 – GetProperty
	flags	0x00
	reserved	0x00
	ParameterCount	0x02
	PropertyTag	0x00000001 - CurrentVersion
	memory ID	0x00000000 - Internal Flash

8.4.3 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 51. CRC16 algorithm

Characteristic	Value
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;
}
```

8.4.4 Ping packet

The ping packet is the first packet sent from a host to the target to establish a connection on the selected peripheral. If the selected peripheral is UART, this packet is used to run autobaud detection. The ping packet can be sent from host to target at any time that the target expects 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 (see [Section 8.4.5](#)). A ping packet is a special type of framing packet (as described in [Section 8.4.2](#)).

Table 52. Ping packet format

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

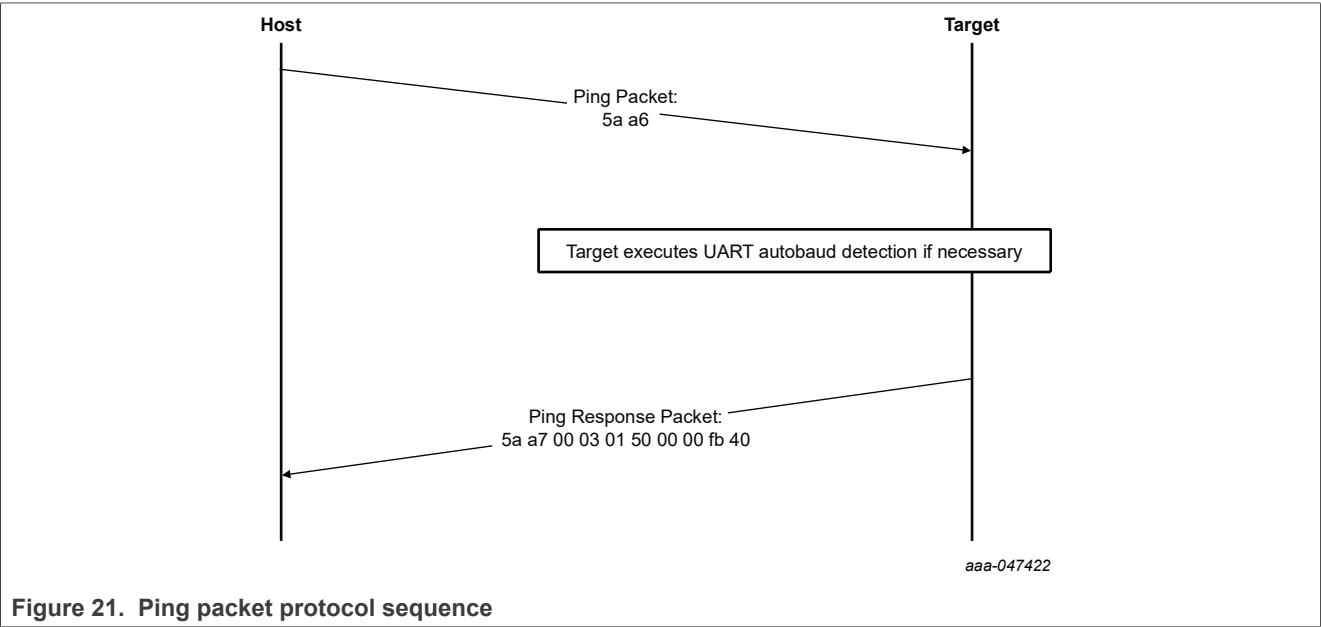


Figure 21. Ping packet protocol sequence

8.4.5 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. When the host receives the ping response packet, the connection is established over this interface. The host starts sending commands to the target.

Table 53. 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, see Section 8.4.3
9	0x40	CRC16 high, see Section 8.4.3

For the UART peripheral, to run autobaud, the host must send a ping packet when a connection is first established. 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 the GetProperty command (see [Section 8.5.2](#) for more details) returns.

8.4.6 Command packet

The command packet carries a 32-bit command header and a list of 32-bit parameters.

Table 54. Command packet format

Command packet format (up to 32 bytes)										
Command header (4 bytes)				28 bytes for parameters (maximum 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 55. 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; must 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 has a maximum size of 32 bytes, only seven parameters can fit into it.

The target also uses command packets 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 56. Command tags

Command tag	Name	Description
0x01	FlashEraseAll	The command tag specifies one of the commands that the bootloader supports. The valid command tags for the bootloader are listed here. Note: Only the <i>GetProperty</i>, <i>Reset</i>, <i>KeyProvisioning</i>, <i>SetProperty</i>, and <i>ReceiveSbFile</i> are allowed in limited ISP mode (CMPA_DIGEST written).
0x02	FlashEraseRegion	
0x03	ReadMemory	
0x04	WriteMemory	
0x05	FillMemory	
0x06	rReserved	
0x07	GetProperty	
0x08	ReceiveSbFile	
0x09	Execute	
0x0A	rReserved	
0x0B	Reset	
0x0C	SetProperty	
0x0D	reserved	
0x0E	reserved	
0x0F	reserved	
0x10	reserved	
0x11	ConfigureMemory	
0x12	reserved	
0x13	reserved	
0x14	reserved	
0x15	KeyProvisioning	

Table 57. 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 (used for sending responses to ReadMemory command only)	
0xA7	GetPropertyResponse (used for sending responses to GetProperty command only)	
0xB5	KeyProvisioningResponse	

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, data packets follow the command sequence. A command-specific parameter in the parameters array determines the number of bytes that are transferred in the data phase.

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.

8.4.7 Response packet

The responses are carried using the same command packet format wrapped with framing packet data, as explained in [Section 8.4.6](#). Types of responses include:

- GenericResponse.
- GetPropertyResponse.
- ReadMemoryResponse.
- KeyProvisioningResponse.

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 58. 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. See Section 8.5.16 for a list of possible status codes.
4 - 7	Command tag	The Command tag parameter identifies the response to the command sent by the host.

GetPropertyResponse: The target sends the GetPropertyResponse packet 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).

To include the status code and one or many property values at all times, the parameter count field in the header is set to greater than 1.

Table 59. 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 target sends the ReadMemoryResponse packet 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 60. 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.

KeyProvisioningResponse: The target sends The KeyProvisioningResponse packet in response to the host sending a KeyProvisioning command. The KeyProvisioningResponse packet contains the framing packet data and command packet data with the command/response tag set to a KeyProvisioningResponse tag value (0xB5) and the flags field set to kCommandFlag_HasDataPhase (1).

Table 61. KeyProvisionResponse parameters

Byte #	Value	Parameter
0 - 3	-	Status code
4 - 7	-	Data Byte count

8.4.8 Data packet

The structure of a data packet is not defined.

A data packet is always used as part of a data phase of the preceding command or response packet. The structure of the data packet is specific to this data phase. It is described in the specific sections for each command (if applicable).

8.4.9 ACK packet

The ACK packet is sent in response to a received command packet, response packet, or data packet. There are 3 kinds of ACK packets:

- Ack packet: The previous packet was received successfully. The sending of more packets is allowed.
- Nak packet: The previous packet was corrupted and must be resent.
- AckAbort packet: Data phase is aborted.

An ACK packet is a special type of framing packet (as described in [Section 8.4.2](#)).

Table 62. Ack packet format

Byte #	Value	Name
0	0x5A	Start byte
1	0xA1	Ack

Table 63. Nak packet format

Byte #	Value	Name
0	0x5A	Start byte
1	0xA2	Nak

Table 64. AckAbort packet format

Byte #	Value	Name
0	0x5A	Start byte
1	0xA3	AckAbort

8.5 Bootloader command set

8.5.1 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 the bootloader returns, see [Section 8.5.16](#).

For a list of command and response tags the bootloader uses, see [Section 8.5.15](#).

8.5.2 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 65. Parameters for GetProperty Command

Byte #	Parameter
0 - 3	property tag See Section 8.5.14 for more details.
4 - 7	external memory identifier (only applies to get property for external memory, see Section 8.5.11.1) or status identifier (if the property tag is equal to 8).

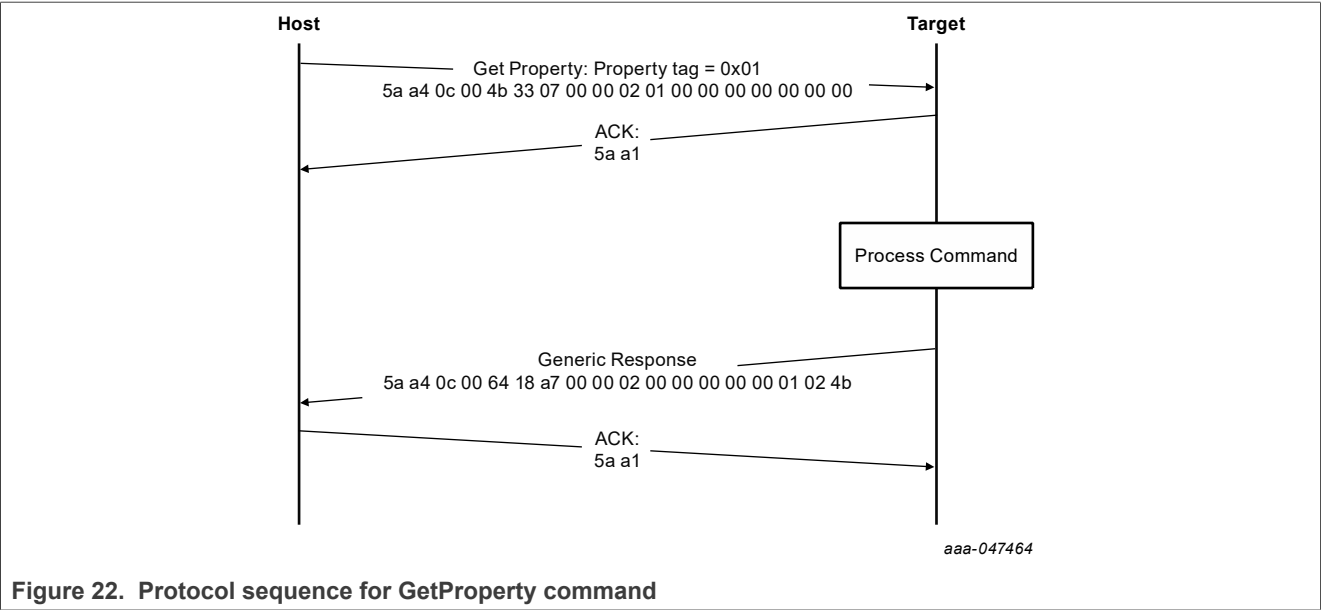


Figure 22. Protocol sequence for GetProperty command

Table 66. GetProperty command packet format (example)

GetProperty	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4 – kFramingPacketType_Command
	Length	0x0C 0x00
	Crc16	0x4B 0x33
Command packet	CommandTag	0x07 – GetProperty
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x02
	PropertyTag	0x00000001 - CurrentVersion
	Memory ID	0x00000000 - Internal Flash

The GetProperty command has no data phase.

Response: In response to a GetProperty command, the target sends a GetPropertyResponse packet with the response tag set to 0xA7. The parameter count indicates the number of parameters sent for the property values, with the first parameter showing status code 0, followed by the property values. [Table 67](#) shows an example of a GetPropertyResponse packet.

Table 67. 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

[Table 68](#) gives the possible status codes.

Table 68. GetProperty response status codes

Status code	Description
kStatus_Success	The command competed successfully.
kStatus_UnknownProperty	This device does not support the property tag.

8.5.3 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 (see [Section 8.5.14](#)). 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 69. Parameters for SetProperty command

Byte #	Command
0 - 3	Property tag See Section 8.5.14 for more details
4 - 7	Property value

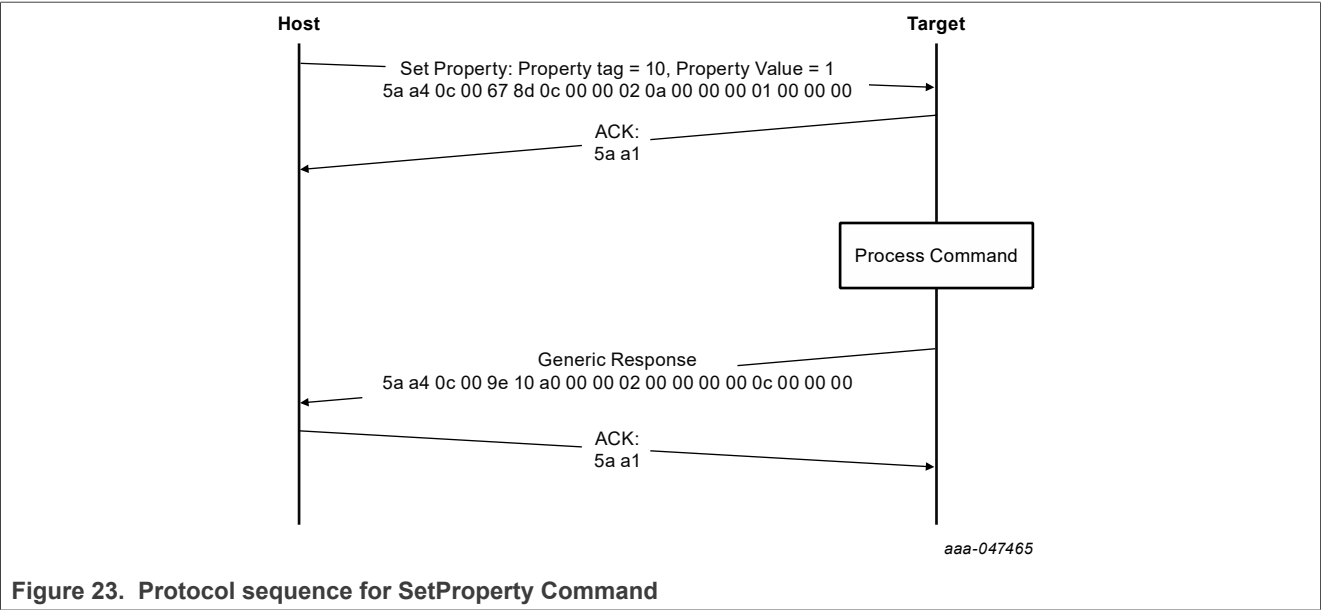


Figure 23. Protocol sequence for SetProperty Command

Table 70. 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 the status codes given in [Table 71](#).

Table 71. SetProperty response status codes

Status code	Description
kStatus_Success	The command completed successfully.
kStatus_ReadOnlyProperty	The property tag
kStatus_UnknownProperty	This device does not support the property tag.
kStatus_InvalidPropertyValue	The property value is not valid for this property tag.

8.5.4 FlashEraseAll command

The FlashEraseAll command performs an erase of the entire flash memory. If any flash regions are protected, 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 (see [Section 8.5.11.1](#)). If memory ID is not specified, the internal flash (memory ID =0) is selected as default.

Note: This command is only available in NXP closed and Tier 1 development lifecycle states. For more information, see [Section 10.6](#).

Table 72. Parameter for FlashEraseAll command

Byte #	Parameter
0-3	Memory ID (see Section 8.5.11.1)

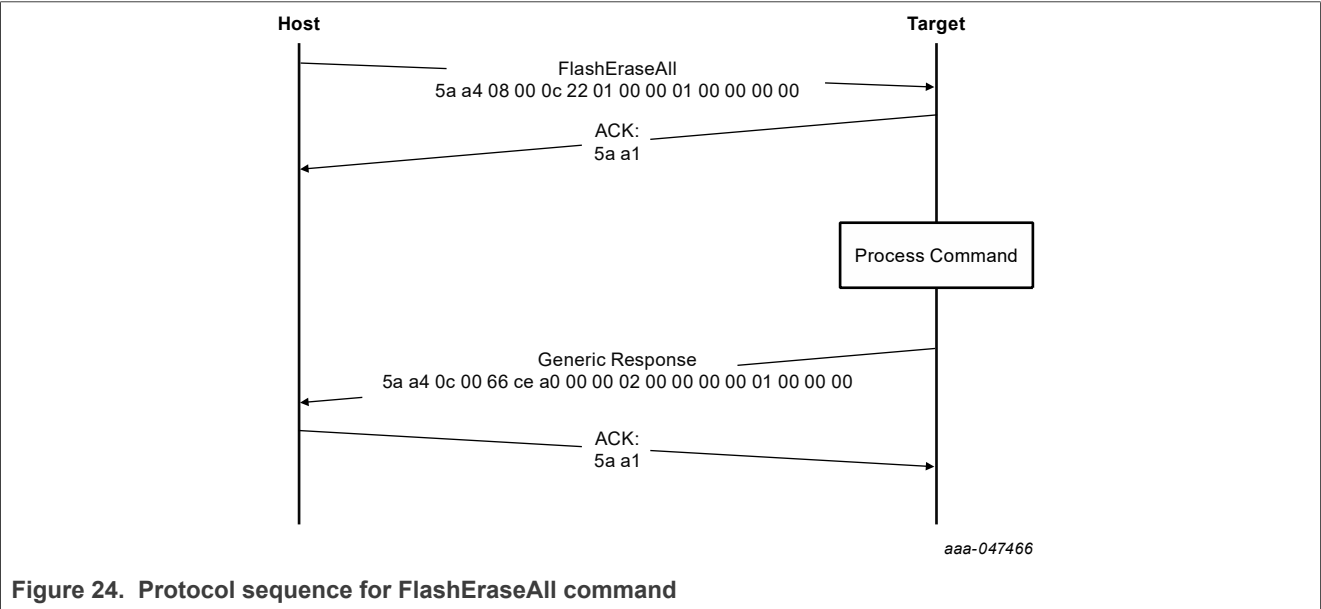


Figure 24. Protocol sequence for FlashEraseAll command

Table 73. 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	See Section 8.5.11.1

The FlashEraseAll command has no data phase.

Response: The target returns a GenericResponse packet with one of the status codes given in [Table 74](#).

Table 74. FlashEraseAll response status codes

Status code	Description
kStatus_Success	The command completed successfully.
kStatus_InvalidArgument	This device does not support the Memory ID.
kStatus_CommandUnsupported	The flash cannot be erased.
kStatus_FLASH_CommandFailure	The flash-erase verification failed (only done when the property VerifyWrites is set to 1).
kStatusMemoryVerifyFailed	The flash-erase verification failed (only done for external flash).

8.5.5 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).

Note: This command is only available in NXP closed and Tier 1 development lifecycle states, see [Section 10.6](#) for more information.

Table 75. Parameter for FlashEraseRegion command

Byte #	Parameter
0-3	Start address
4 - 7	Byte count
8 - 11	Memory ID (see Section 8.5.11.1)

The FlashEraseRegion command has no data phase.

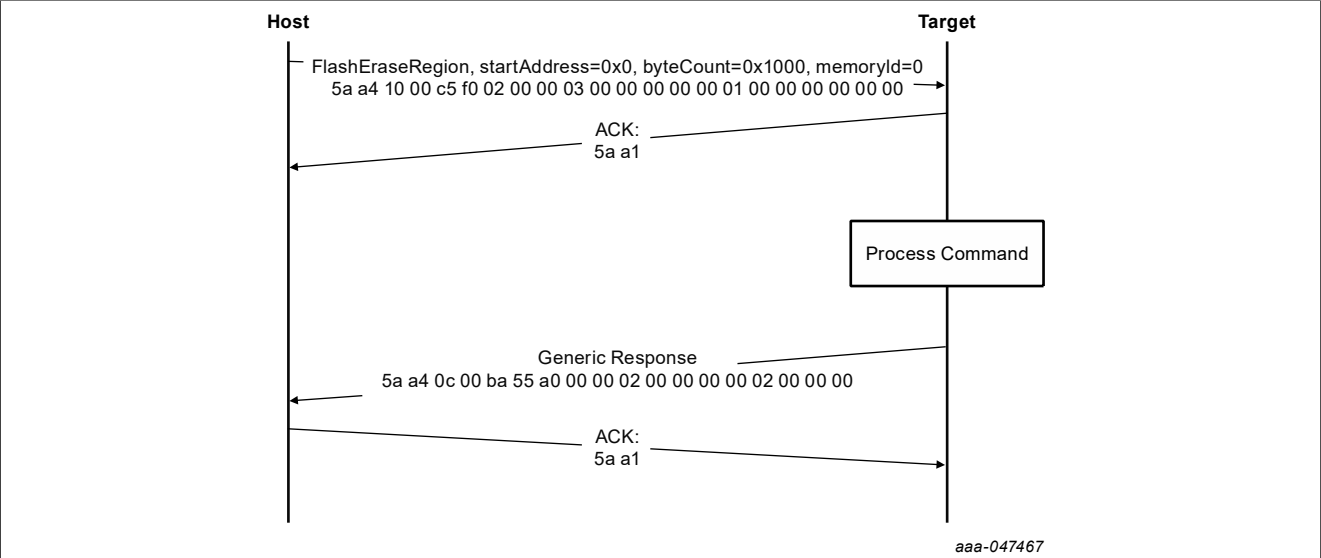


Figure 25. Protocol sequence for FlashEraseRegion command

Response: The target returns a GenericResponse packet with one of the error status codes given in [Table 76](#).

Table 76. FlashEraseRegion response status codes

Status code	Description
kStatus_Success	The command completed successfully.
kStatus_Fail	The specified region overlaps a PRINCE-encrypted region that is not fully contained in the specified region.
kStatus_MemoryRangeInvalid	The specified region overlaps a reserved region or this device does not support the Memory ID.
kStatus_InvalidArgument	The specified region falls outside the flash (only for external flash).
kStatus_FLASH_AlignmentError	The start address and/or length are not aligned to the flash sector size (8192 bytes).
kStatus_FLASH_AddressError	The specified region falls outside the flash (only for internal flash).
kStatus_FLASH_CommandFailure	The flash-erase verification failed (only done when the property VerifyWrites is set to 1).
kStatusMemoryVerifyFailed	The flash-erase verification failed (only done for external flash).

8.5.6 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 (see [Section 8.5.11.1](#)) is optional. If the memory ID is not specified, internal memory is selected as default.

Note: This command is only available in NXP closed and Tier 1 development lifecycle states. For more information, see [Section 10.6](#).

Table 77. 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, see Section 8.5.11.1 .

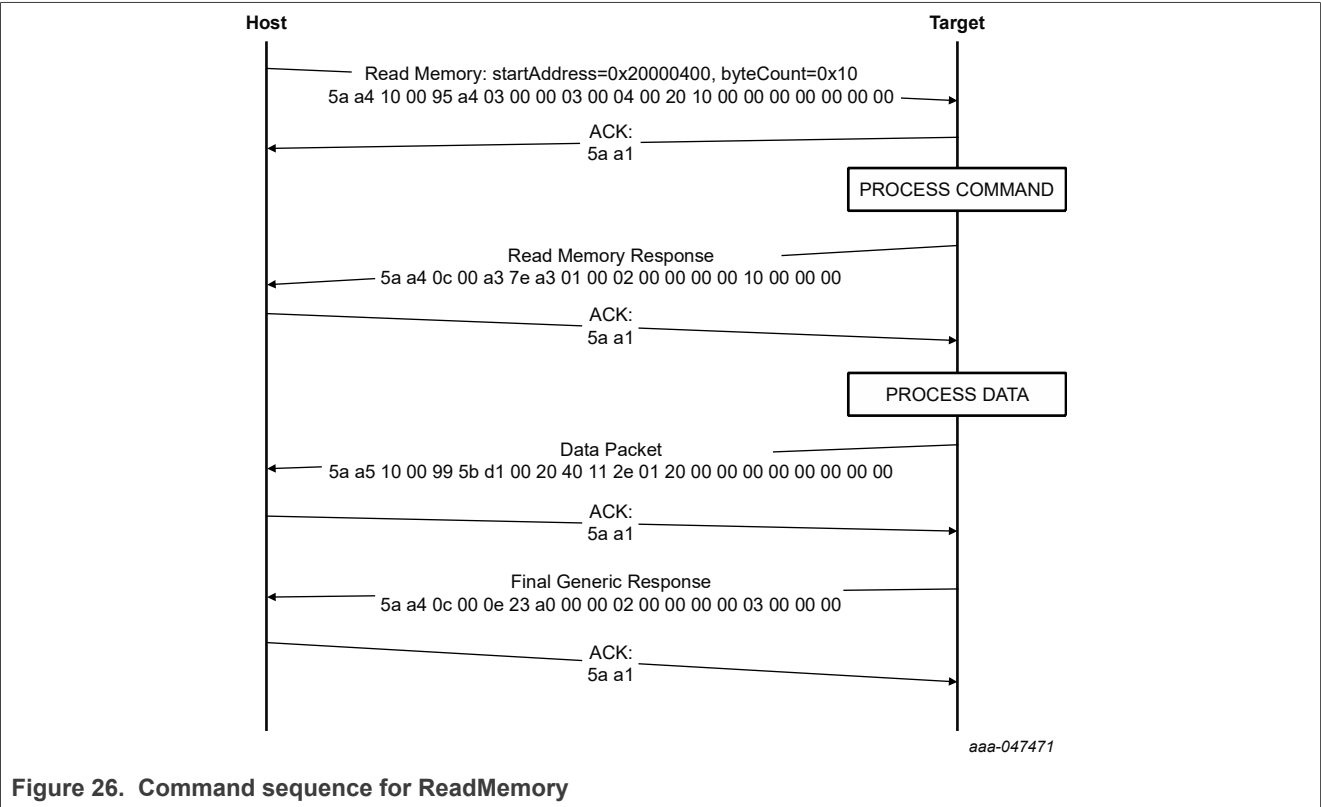


Figure 26. Command sequence for ReadMemory

Table 78. ReadMemory command packet format (example)

ReadMemory	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	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 must pull data packets until the host receives the number of bytes of data specified in the byteCount parameter of ReadMemory command.

Response: The target returns a GenericResponse packet with one of the error status codes given in [Section 8.5.6](#).

Table 79. ReadMemory response status codes

Status code	
kStatus_Success	The command completed successfully.
kStatusMemoryBlankPageReadDisallowed	An internal flash page that is still entirely erased cannot be read.
kStatusMemoryProtectedPageReadDisallowed	PRINCE-encrypted regions are protected and cannot be read.
kStatus_MemoryRangeInvalid	The specified region overlaps a reserved region or this device does not support the Memory ID.
kStatus_InvalidArgument	The specified region falls outside the flash.
kStatus_FLASH_AlignmentError	An attempt is done to read in the protected flash regions outside the specified fields.
kStatus_FLASH_NmpaAccessNotAllowed	The NMPA is protected and cannot be read.

8.5.7 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, 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 command.
- Writing to flash requires the start address to be phrase (16-byte) aligned.
- The byte count is rounded up to a phrase size and trailing bytes are filled with 0x00.

- If the VerifyWrites property is set to true, writes to flash and also performs a flash verify program operation.

When writing to RAM, the start address does not have 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 (see [Section 8.5.11.1](#)) is optional. If memory ID is not specified, internal memory is selected as default.

Note: This command is only available in NXP closed and Tier 1 development lifecycle states. For more information, see [Section 10.6](#).

Table 80. Parameters for WriteMemory command

Byte #Command	
0-3	Start address
4-7	Byte count
8-11	Memory ID (see Section 8.5.11.1)

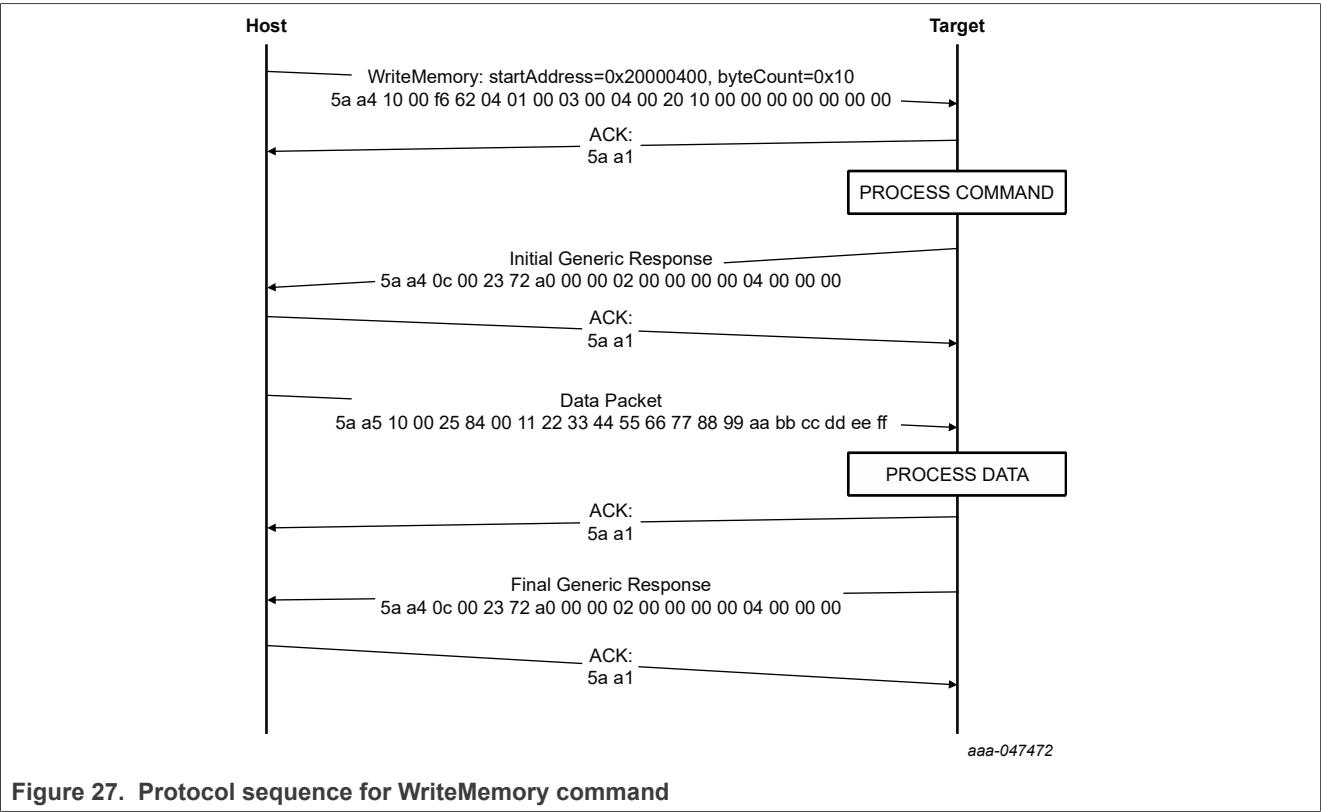


Figure 27. Protocol sequence for WriteMemory command

Table 81. 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 target receives the number of bytes of data specified in the byteCount parameter of the WriteMemory command.

Response: The target returns a GenericResponse packet with one of the error status codes given in [Table 82](#).

Table 82. WriteMemory response status codes

Status code	Description
kStatus_Success	The command completed successfully.
kStatus_Fail	The specified region overlaps a PRINCE-encrypted region that is not fully contained in the specified region.
kStatusMemoryCumulativeWrite	The specified region (in internal flash) is not erased.
kStatusMemoryRangeInvalid	The specified region overlaps a reserved region or the device does not support the Memory ID.
kStatus_FLASH_CommandFailure	A hardware error occurred when writing to internal flash.
kStatus_FLASH_AddressError	The specified region falls outside the internal flash.
kStatus_FLASH_AlignmentError	Address or length is not properly aligned for writing to internal flash.
kStatus_FLASH_CompareError	The flash program verification failed (only done when the property VerifyWrites is set to 1).
kStatus_FLASH_InvalidArgument	An attempt is done to write in the protected flash regions outside the specified fields or an attempt is done to write the Key Store while property FfrKeystoreUpdateOpt is different from 1.
kStatus_FLASH_FfrBankIsLocked	The protected flash region is locked and cannot be written anymore.
kStatus_FLASH_SizeError	The CFPA scratch page must be written completely at once.
kStatus_FLASH_OutOfDateCfpaPage	The CFPA scratch page must have a higher version number than the currently active CFPA page.
kStatus_FLASH_SealedFfrRegion	The CMPA page digest is already written so the page cannot be written anymore.

8.5.8 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 the FillMemory command parameter. Also, there is no data phase for the FillMemory command, while WriteMemory does have a data phase.

Note: This command is only available in NXP closed and Tier 1 development lifecycle states. For more information, see [Section 10.6](#).

Table 83. 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 a multiple of 4.
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 the 32-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.
- Writing to flash requires the start address to be page (128-byte) aligned.
- If the VerifyWrites property is set to true, writes to flash and also performs a flash verify program operation.

When writing to RAM, the start address does not have to be aligned.

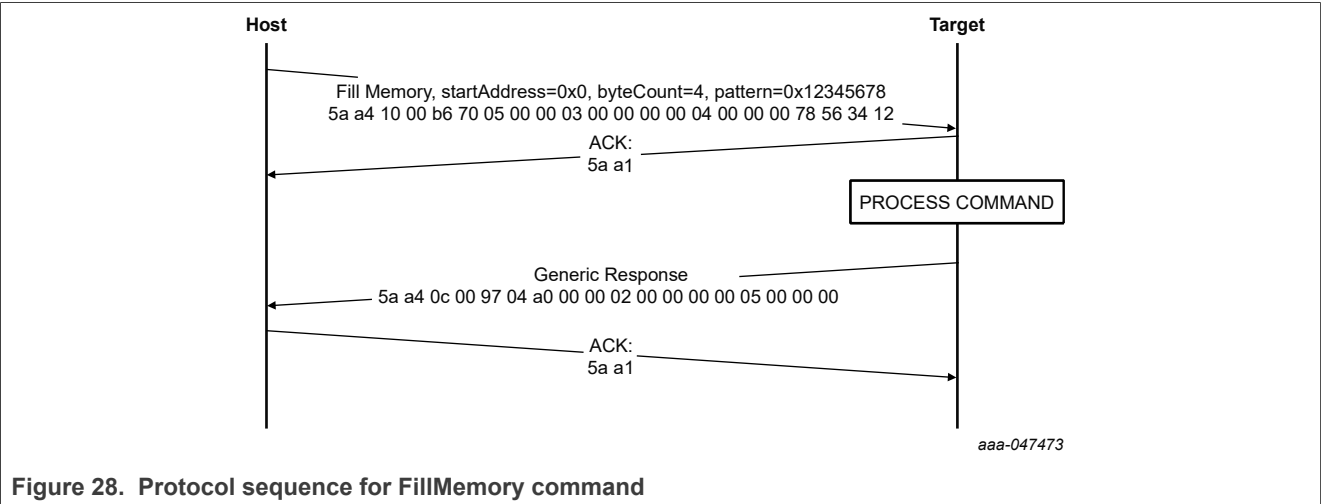


Figure 28. Protocol sequence for FillMemory command

Table 84. 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: The target returns a GenericResponse packet with one of the error status codes given in [Table 85](#).

Table 85. FillMemory response status codes

Status code	Description
kStatus_Success	The command completed successfully.
kStatusMemoryCumulativeWrite	The specified region (in internal flash) is not erased.
kStatus_FLASH_CommandFailure	A hardware error occurred when writing to internal flash.
kStatusMemoryUnsupportedCommand	The specified region falls outside the internal flash or RAM.
kStatus_FLASH_AlignmentError	Address or length is not properly aligned for filling internal flash.
kStatus_FLASH_CompareError	The flash program verification failed (only done when the property VerifyWrites is set to 1).

8.5.9 Execute command

The execute command results in the bootloader setting the program counter to the code at the provided jump address and a stack pointer to the provided stack pointer address. Before the jump, the system is returned to the reset state. The load address is the base address where the first word of the image is present in memory.

The jump address, load address (64-byte aligned), 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.

Note: This command is only available in NXP closed and Tier 1 development lifecycle states. For more information, see [Section 10.6](#).

Table 86. Parameters for execute command

Byte #	Command
0-3	Jump address
4-7	Load address

Table 86. Parameters for execute command...continued

Byte #	Command
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 one of the error status codes given in [Table 87](#).

Table 87. Execute response status codes

Status code	Description
kStatus_Success	The command completed successfully.
kStatus_CommandUnsupported	The parameters are not correct and the image is not executed.

8.5.10 Reset command

The reset command results in the bootloader resetting the chip.

The reset command requires no parameters.

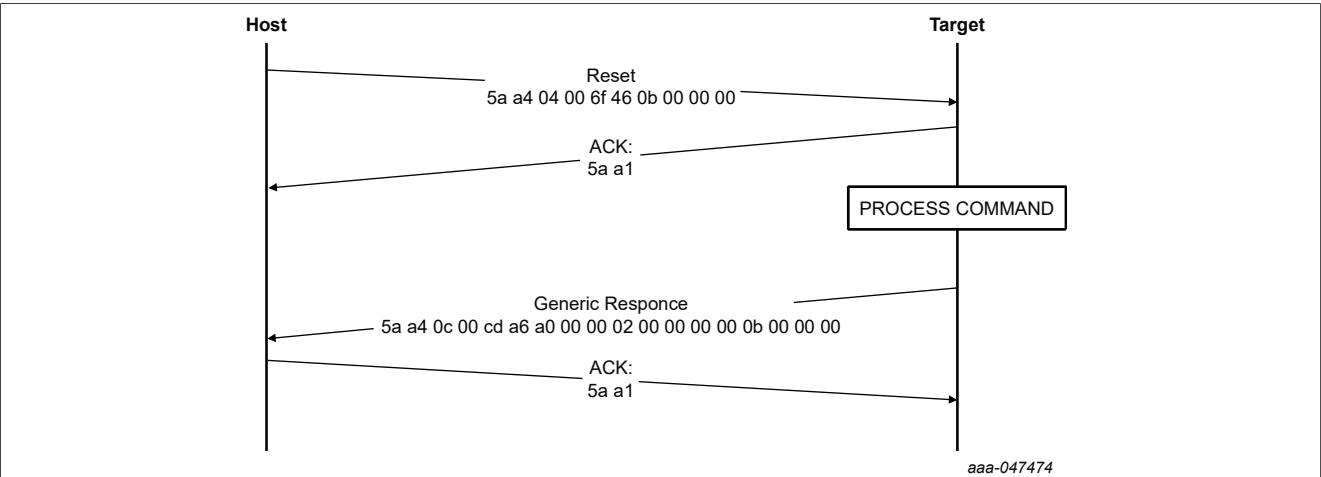


Figure 29. Protocol sequence for reset command

Table 88. 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	0x00

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 trigger a boot from flash after successful flash image provisioning via the ROM bootloader. After issuing the reset command, allow five seconds for the user application to start running from Flash.

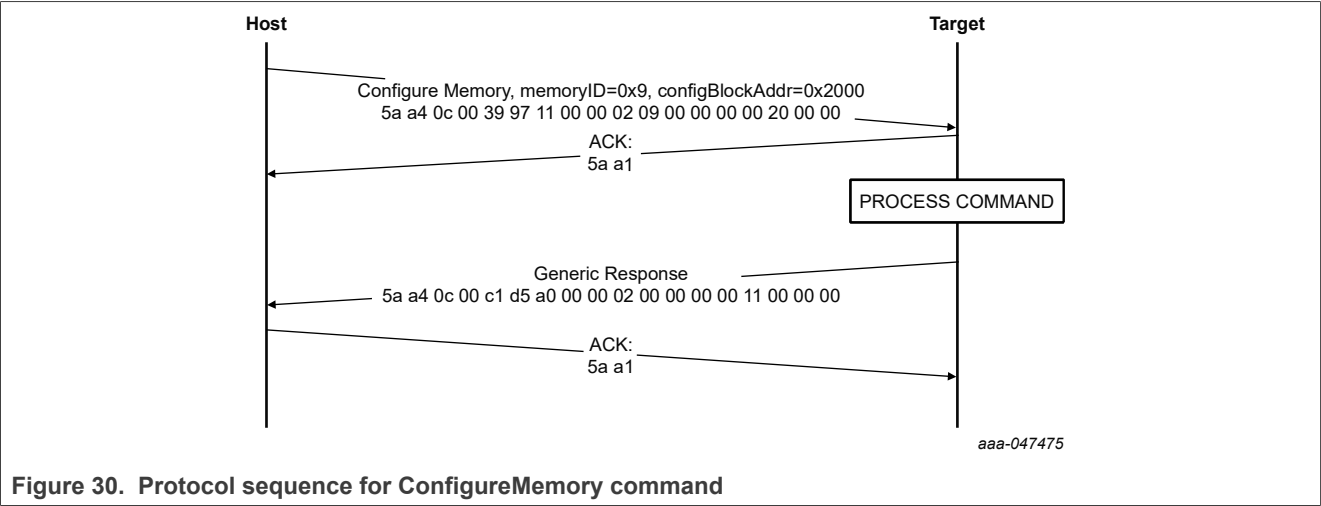
8.5.11 ConfigureMemory command

The ConfigureMemory command configures the internal/external memory device using a preprogrammed configuration block. The parameters passed in the command are the memory ID (see [Section 8.5.11.1](#)) and then the memory address 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 location. Then this command directs the bootloader to use the data at that location to configure the external memory devices. An example of configuration data for an external NOR flash device is in [Section 8.5.11.2](#).

Note: This command is only available in NXP closed and Tier 1 development lifecycle states. For more information, see [Section 10.6](#).

Table 89. Parameters for ConfigureMemory command

Byte #	Command
0-3	Memory ID, see Section 8.5.11.1
4-7	Configuration block address



Response: The target returns a GenericResponse packet with one of the error status codes given in [Table 90](#).

Table 90. ConfigureMemory response status codes

Status code	Description
kStatus_Success	The command completed successfully.
kStatus_Fail	The PRINCE encryption configuration could not be applied (for internal flash).
kStatus_InvalidArgument	The supplied configuration block has invalid parameters or this device does not support the Memory ID.

8.5.11.1 Supported memory IDs

[Table 91](#) shows the supported memory IDs.

Table 91. Supported memory IDs

Memory ID	Description
0	Internal flash (Used for the PRINCE configuration)
0x110	External 1-bit SPI NOR flash device

8.5.11.2 1-bit SPI NOR flash support

The boot ROM supports programming 1-bit SPI NOR flash devices, which support 3-byte address read (command ID 0x03) or 4-byte address read (command ID 0x13), via the following flash configuration option block. An example of the commands required is shown below.

First, the boot ROM uses the read MID (command ID 0x9F) to detect if there is a connected device to the MCX W23.

To determine the flash parameters (such as total size, sector size, ...), the boot ROM supports the manually configured option and the autodetected option (using the read SFDP command ID 0x5A).

When using the manually configured option, users must specify all the flash information (flash size, sector size, page size) in the option block. When using the autodetected option, the boot ROM can detect the flash information via the read SFDP (0x5A) command, where users can set all the flash information to 0x0s.

[Table 92](#) 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 exceeds 16 MB (detected by the read SFDP command), the 0x13 command is used for page read.

Note: The bootloader uses Flexcomm 2 to access the 1-bit SPI NOR flash (on different pins than the UART ISP interface). It means that UART cannot be used as ISP interface for commands using the 1-bit SPI NOR flash. UART can still be used as ISP interface for all other commands, even when a 1-bit SPI NOR flash is also connected.

Table 92. Serial NOR flash configuration option block

Field	Tag	Reserved	Flash info set	Flash size	Sector size	Page size
	[31:28]	[27:16]	[15:12]	[11:8]	[7:4]	[3:0]
option0	0xc	-	0 - Manual (select flash parameter via flash data sheet)	0 = 512 KB	0 = 4 KB	0 = 256 bytes
				1 = 1 MB	1 = 8 KB	1 = 512 bytes
				2 = 2 MB	2 = 32 KB	2 = 1 KB
			1 - Reserved. Do not use	3 = 4 MB	3 = 64 KB	3 = 28 KB
				4 = 8 MB	4 = 128 KB	-
				5 = 16 MB	5 = 256 KB	
			2 - Auto (detect flash parameter via SFDP table)	6 = 32 MB	-	
				7 = 64 MB		
				8 = 64 MB		
				9 = 128 MB		
				10 = 256 MB		

Example of programming 1-bit SPI NOR flash via boot ROM

This example uses the autodetected parameter. The configuration option block is 0xc000_2000 then. Here are the steps to enable the 1-bit SPI NOR flash programming using the boot ROM:

```
# Fill the configuration option block to RAM address 0x2001_0000
blhost -p COM12 -b i2c,0x10 -- fill-memory 0x20010000 4 0xc0002000
# Enable 1-bit SPI NOR FLASH support using configuration option block stored at
0x2001_0000
blhost -p COM12 -b i2c,0x10 -- configure-memory 0x110 0x20010000
# Erase the entire external flash
blhost -p COM12 -b i2c,0x10 -- flash-erase-all 0x110
# Program boot image to the 1-bit SPI NOR FLASH at address 0 (start of the
external flash)
blhost -p COM12 -b i2c,0x10 -- write-memory 0 <boot_image> 0x110
```

8.5.12 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 the bootloader receives it. For more information about the SB file, see [Section 7.3.6](#).

Table 93. 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 target receives the number of bytes of data specified in the byteCount parameter of the receive SB file command.

Response: The target returns a GenericResponse packet with one of the error status codes given in [Table 94](#).

Table 94. ReceiveSBFile response status codes

Status code	Description
kStatus_Success	The command completed successfully.
kStatus_Fail	An unspecified error occurred while processing the SB file.
kStatusRomLdrSignature	The signature or the SB file version is incorrect.
kStatusRomLdrSectionLength	The section chunk count is incorrect.
kStatusRomLdrRollbackBlocked	The user build number, non-secure firmware version or secure firmware version is smaller than expected.
kStatusRomLdrEOFReached	The SB file size is smaller than expected.
kStatusRomLdrUnexpectedCommand	The SB file contains an unexpected or unsupported command.
kStatusRomLdrUnknownCommand	The SB file contains an unknown command.
kStatusRomLdrChecksum	The SB file contains a command with an incorrect checksum.
kStatusRomLdrInvalidSectionMacCount	The section MAC count is incorrect.
kStatusRomLdrIdNotFound	The SB file does not contains a valid boot section.
kStatusRomLdrCrc32Error	The SB file contains an incorrect checksum.
kStatus_InvalidArgument	The flash erase command contains invalid parameters.

8.5.13 KeyProvisioning command

The KeyProvisioning command is a pack of several security-related commands, to install preshared keys, generate random keys, and save them into the protected flash region - customer key store area.

There are three parameters for the KeyProvisioning command. They are listed in [Table 95](#). The first parameter, <Key Operation> is required to specify the KeyProvisioning command behavior. The other two parameters, <Key Type> and <Key Size> are required for certain KeyProvisioning operations.

Table 95. 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 96](#) and [Table 97](#) describe the details of each Key operation and Key Type.

Table 96. Key operation details

Value	Operation	Details
1	SetUserKey	Send <Key size> bytes of the <Key Type> key to ROM from host. Incoming data Phase is required to transfer the key bytes.
2	SetIntrinsicKey	Generate <Key Size> bytes of the key specified by <Key Type> in key store
3	WriteNonVolatile	Write the key store in RAM to a nonvolatile memory specified by <Memory ID>. Only Memory ID 0 (internal flash) is supported. <Key Size> is not used for this operation
4	ReadNonVolatile	Load the key store to RAM from a nonvolatile memory specified by <Memory ID>. Only Memory ID 0 (internal flash) is supported. <Key Size> is not used.
5	WriteKeyStore	Send the key store to RAM 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 RAM 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 97. Key Type details

Value	Key Type
0x0	Invalid
0x1	Reserved
0x2	Reserved
0x3	Firmware update key 0 (SBKEK)
0x4	Reserved
0x5	Reserved
0x6	Reserved
0x7	Encrypted region key 0 (PRINCE)
0x8	Encrypted region key 1 (PRINCE)
0x9	Encrypted region key 2 (PRINCE)

Table 97. Key Type details...continued

Value	Key Type
0xA	Encrypted region key 3 (PRINCE)
0xB	User key
0xC	UDS key

Command: The KeyProvisioning command packet format is shown in [Table 98](#).

Table 98. KeyProvisioning 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
	reserved	0x00
	parameter count	0x03
	key operation	0x00000002 (see Table 96)
	key type / memory ID	0x00000000 (see Table 97)
	key size	0x00000100

Data phase: Determined by <Key Operation> based on the Incoming or outgoing data phase of the KeyProvisioning command.

For an incoming packet, the host sends data packets until <Key Size> specifies the number of data bytes or the target receives the key store size.

For outgoing data phase, the host must pull data packets until it receives the entire key store data bytes. The KeyProvision response sends the key store size to the host.

Response: The target returns a GenericResponse packet for the key operations without data phase. It returns a KeyProvisioningResponse packet for the other key operations, such as WriteKeyStore.

For the GenericResponse, see [Section 8.4.7](#).

[Table 61](#) describes the KeyProvisionResponse packet.

Table 99. KeyProvisioning 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

[Table 100](#) gives the error status codes that can be returned.

Table 100. KeyProvisioning response status codes

Status code	
kStatus_Success	The command completed successfully.
kStatus_Fail	An unspecified error occurred while processing command.

8.5.14 Get/SetProperty command properties

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

Table 101. Properties used by Get/SetProperty commands, sorted by values

Property	Writable	Tag Value	Size	Description
CurrentVersion	No	01h	4	Current bootloader version. Value returned on MCX W23xB is 0x4B030100. This is K3.1.0 in ASCII
AvailablePeripherals	No	02h	4	The set of peripherals supported on this chip. Value returned for MCX W23xB is 0x7 - indicating that UART, I ² C slave and SPI slave is supported
FlashStartAddress	No	03h	4	Start address of program flash.
FlashSizeInBytes	No	04h	4	Size in bytes of program flash.
FlashSectorSize	No	05h	4	Size in bytes of a program flash sector.
AvailableCommands	No	07h	4	The set of commands supported by the bootloader Value returned by MCX W23xB is 0x110 DDF meaning that following commands are supported - FlashEraseAll - FlashEraseRegion - ReadMemory - WriteMemory

Table 101. Properties used by Get/SetProperty commands, sorted by values...continued

Property	Writable	Tag Value	Size	Description
				• FillMemory • GetProperty • ReceiveSBFile • Execute • Reset • SetProperty • ConfigureMemory • KeyProvisioning
Check Status	No	08h	4	Return the status based on specified status identifier Status = 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 Status = 1 - Last Error See the details of last error
VerifyWrites	Yes	0Ah	4	If the value of this property is 1, after each flash erase or flash write, the contents of the flash is verified. If the value of this property is 0, no verification is done.
MaxPacketSize	No	0Bh	4	Maximum supported packet size for the currently active peripheral interface.
ReservedRegions	No	0Ch	8*n	List of memory regions the bootloader reserves. Returned as value pairs (<start-address-of-region>, <end-address-of-region>). If HasDataPhase flag is not set, the response packet parameter count indicates the number of pairs. If HasDataPhase flag is set, the second parameter is the number of bytes in the data phase. "n" indicates number of memory region pairs
RamStartAddress	No	0Eh	4	Start address of RAM
RamSizeInBytes	No	0Fh	4	Size in bytes of RAM
SecurityState	No	11h	4	Indicates whether Flash security is enabled 0x5aa55aa5 - Device is in development lifecycle 0xc33cc33c - Device is in deployment lifecycle
UniqueDevice/UUID	No	12h	16	Unique device identification
TargetVersion	No	18h	4	Target version Value returned for MCX W23xB is 0x54010000. This is T1.0.0 in ASCII
ExternalMemoryAttributes	No	19h	24	List of attributes the specified memory Id (0x110 = SPI NOR flash) supports. See description for the return value in the section ExternalMemoryAttributes Property.

Table 101. Properties used by Get/SetProperty commands, sorted by values...continued

Property	Writable	Tag Value	Size	Description
FlashPageSize	No	1Bh	4	Size in bytes of a program flash page.
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
FfrKeystoreUpdateOpt	Yes	1Dh	4	Supported key store update methods 0x0 - key store can only be updated with Key Provisioning command 0x1 - key store can be updated with Key Provisioning and WriteMemory command

Property Definition:

Most Get/Set properties are a 32-bit word that is interpreted as a number. This section gives details about the properties that have a defined structure.

- CurrentVersion property**

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

Table 102. CurrentVersion property fields

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

- AvailablePeripherals property**

The value of this property is a bit field that lists the peripherals supported by the bootloader and the hardware on which it is running.

Table 103. Peripheral bits

Bit	[31:7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Field	Reserved	Reserved	Reserved	Reserved	Reserved	SPI Slave	I2C Slave	UART

- AvailableCommands property**

This property value is a bit field 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 bit field. 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: $\text{mask} = 1 \ll (\text{tag} - 1)$.

Table 104. Command bits

Bit	[Others]	[20]	[16]	[15]	[14]	[13]	[12]
Command	Reserved	Key Provisioning	Configure Memory	FlashRead Resource	Flash ReadOne	Flash Program One	FlashErase AllUnsecure
Bit	[11]	[10]	[9]	[8]	[7]	[6]	

Table 104. Command bits...continued

Bit	[Others]	[20]	[16]	[15]	[14]	[13]	[12]
Command	Execute	Receive SBFile	GetProperty	Flash Security Disable	FillMemory	Write Memory	
Bit	[5]	[4]	[3]	[2]	[1]	[0]	
Command	Read Memory	FillMemory	Write Memory	Read Memory	FlashERase Region	Flash EraseAll	

- **ExternalMemoryAttributes property**

The value this property returns is a 24-byte data structure containing available external memory attributes start address, total size in kB, page size, sector size, and block size. [Table 105](#) shows the breakdown of 24-byte structure.

Table 105. 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 bit field set. 0x00000001 - start address 0x00000002 - total size 0x00000004 - page size 0x00000008 - sector size 0x00000010 - block size
4 - 7	Start address of external memory.
8 - 11	Total size of external memory in kilobytes.
12 - 15	Page size of external memory in bytes.
16 - 19	Sector size of external memory in bytes.
20 - 23	Block size of external memory in bytes.

- **GetLastError property**

[Table 106](#) lists the response words and corresponding error conditions.

Table 106. Response word and error description

Response 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.
0x0b36f301	kLog_Auth_Auth_Fail_Time	Image authentication failed. Next word is the duration in microseconds of authentication.
0x0b38f300	kLog_Auth_Dice_Fail	DICE calculation failed.
0x0b39f300	kLog_Auth_BootSeed_Fail	BootSeed calculation failed.
0x0d70f300	kLog_Tzm_DeviceMode_Fail	Unable to determine device TrustZone support.
0x0d71f300	kLog_Tzm_FusesMode_Fail	Unable to determine secure boot Trust Zone image type from CMPA.

Table 106. Response word and error description...continued

Response word	Error	Description
0x0d72f300	kLog_Tzm_ImageMode_Fail	Secure boot TrustZone image type from CMPA is invalid.
0x0c00f301	kLog_Jump_Fail_Reason	The device configuration does not support the image type of the image. Next word is the failing configuration.
0x0c00f500	kLog_Jump_Fail_Fatal	Failed to jump to application.
0x0301f300	kLog_SecureBoot_Init_Fail	Initialization of security-related hardware failed.
0x0801f300	kLog_Ispboot_Init_Fail	Trying to enter ISP mode while no peripherals are enabled for ISP mode.
0x0602f301	kLog_PassiveBoot_Fail_Reason	Passive boot failed. Next word is the error that prevented passive boot.
0x0702f301	kLog_Recoveryboot_Fail_Reason	Recovery boot failed. Next word is the error that prevented recovery boot.

8.5.15 Bootloader command and response tags

This section lists all command tags and all response tags the bootloader uses.

Table 107. Command and response tags

Tag	Tag Value	Command reference
GenericResponse	A0h	See all commands that do not have a dedicated response tag.
FlashEraseAll	01h	See Section 8.5.4 .
FlashEraseRegion	02h	See Section 8.5.5 .
ReadMemory	03h	See Section 8.5.6 .
ReadMemoryResponse	A3h	See Section 8.5.6 .
WriteMemory	04h	See Section 8.5.7 .
FillMemory	05h	See Section 8.5.8 .
GetProperty	07h	See Section 8.5.2 .
GetPropertyResponse	A7h	See Section 8.5.2 .
ReceiveSbFile	08h	See Section 8.5.12 .
Execute	09h	See Section 8.5.9 .
Reset	0Bh	See Section 8.5.10 .
SetProperty	0Ch	See Section 8.5.3 .
ConfigureMemory	11h	See Section 8.5.11 .
KeyProvisioning	15h	See Section 8.5.13 .
KeyProvisioningResponse	B5h	See Section 8.5.13 .

8.5.16 Bootloader status error codes

This section describes the status error codes that the bootloader returns to the host.

Note: In UART, I2C, and SPI ISP modes, the MCX W23 expects that the host reads posted responses within 20 ms * (number of bytes). Otherwise, the MCX W23 reports the abort command. Considering an ACK from the MCX W23 is only two bytes, the host must read those bytes within 40 ms of the MCX W23 posting.

Table 108. Bootloader status error codes, sorted by value

Error Code	Value (decimal)	Description
kStatus_Success kStatus_FLASH_Success kStatus_RomApiExecuteCompleted	0	Operation succeeded without error. The flash operation is successful. ROM successfully completed processing of SB file/boot image.
kStatus_Fail	1	Operation failed with a generic error.
kStatus_ReadOnly	2	Requested value cannot be changed because it is read-only.
kStatus_OutOfRange	3	Requested value is out of range.
kStatus_InvalidArgument	4	Argument of the requested command is undefined.
kStatus_Timeout	5	A timeout occurred.
kStatus_NoTransferInProgress	6	No send in progress.
See Section 9.3	100-199	The flash-specific errors are described in Section 9.3

Table 108. Bootloader status error codes, sorted by value...continued

Error Code	Value (decimal)	Description
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 SB 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 the SB file jumped to has returned.
kStatusRomLdrCallFailed	10111	The call command in the SB file has failed.
kStatusRomLdrKeyNotFound	10112	A matching key was not found in the key dictionary to unencrypt the section of the SB file.
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.
kStatusRomLdrPendingJumpCommand	10119	The jump command is handled when finalizing the processing of the SB file.
kStatusMemoryRangeInvalid	10200	The requested address range does not match an entry. Or the length extends past the end address of the matching entry.
kStatusMemoryReadFailed	10201	Memory read failed.
kStatusMemoryWriteFailed	10202	Memory write failed.
kStatusMemoryCumulativeWrite	10203	Cumulative write occurred due to write to an unerased flash region.
kStatusMemoryNotConfigured	10205	Memory not configured before access.
kStatusMemoryAlignmentError	10206	Alignment error occurred during memory access.
kStatusMemoryVerifyFailed	10207	Verification operation failed after programming/erasing flash.
kStatusMemoryWriteProtected	10208	The memory written to is write-protected.
kStatusMemoryAddressError	10209	Invalid or wrong memory address has been specified.

Table 108. Bootloader status error codes, sorted by value...continued

Error Code	Value (decimal)	Description
kStatusMemoryBlankCheckFailed	10210	Check of blank memory status has failed.
kStatusMemoryBlankPageReadDisallowed	10211	Memory is blank and read command is not allowed.
kStatusMemoryProtectedPageRead Disallowed	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_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.

8.6 UART ISP

8.6.1 Introduction

The bootloader integrates an autobaud detection algorithm for the UART peripheral, providing flexible baud rate choices.

Autobaud feature: If UART is used to connect to the bootloader, the UART_RX pin must be kept high and not left floating during the detection phase to comply with the autobaud detection algorithm. After the bootloader detects the ping packet (0x5A 0xA6) on UART_RX, its firmware executes the autobaud sequence.

If the baud rate is successfully detected, 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 baud rate. 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, the autobaud detection algorithm may calculate an incorrect baud rate. If an incorrect baud rate is calculated, the autobaud detection state machine must 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, 384000, 460800, and 1000000. 384000, 460800 and 1000000 are only supported when using a 32 MHz system clock.

Packet transfer: After autobaud detection has succeeded, 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.

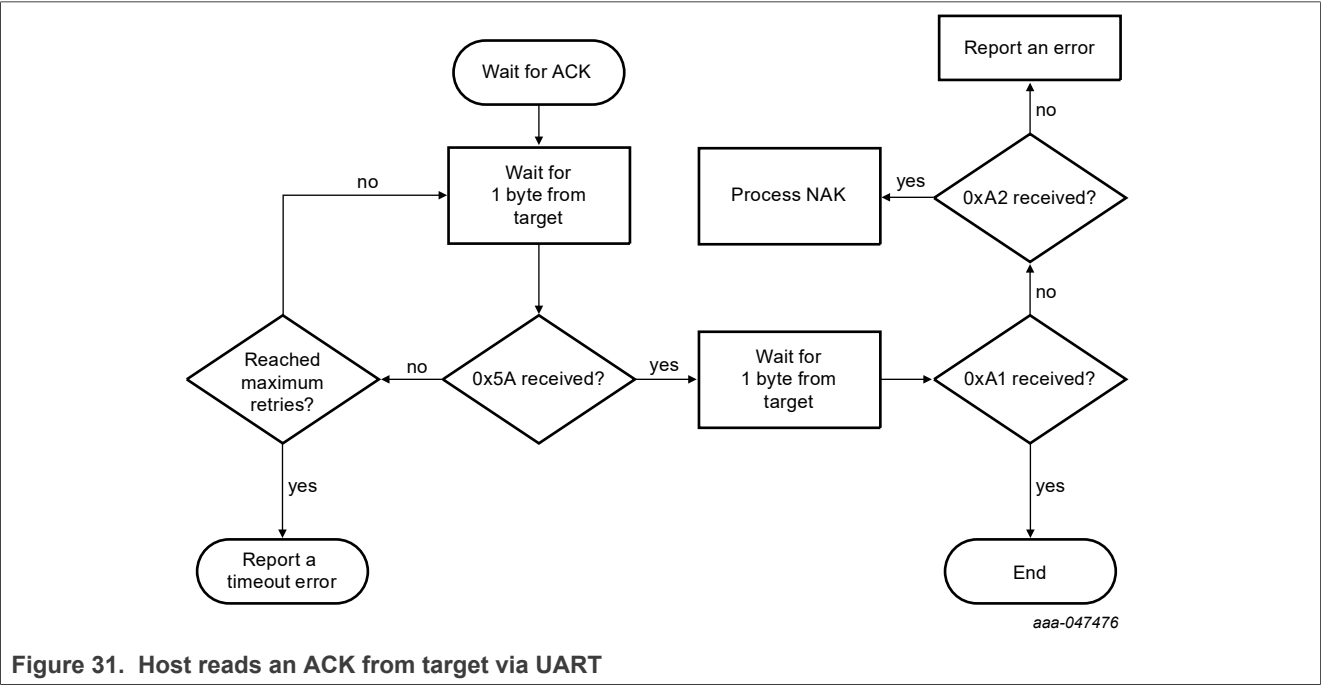


Figure 31. Host reads an ACK from target via UART

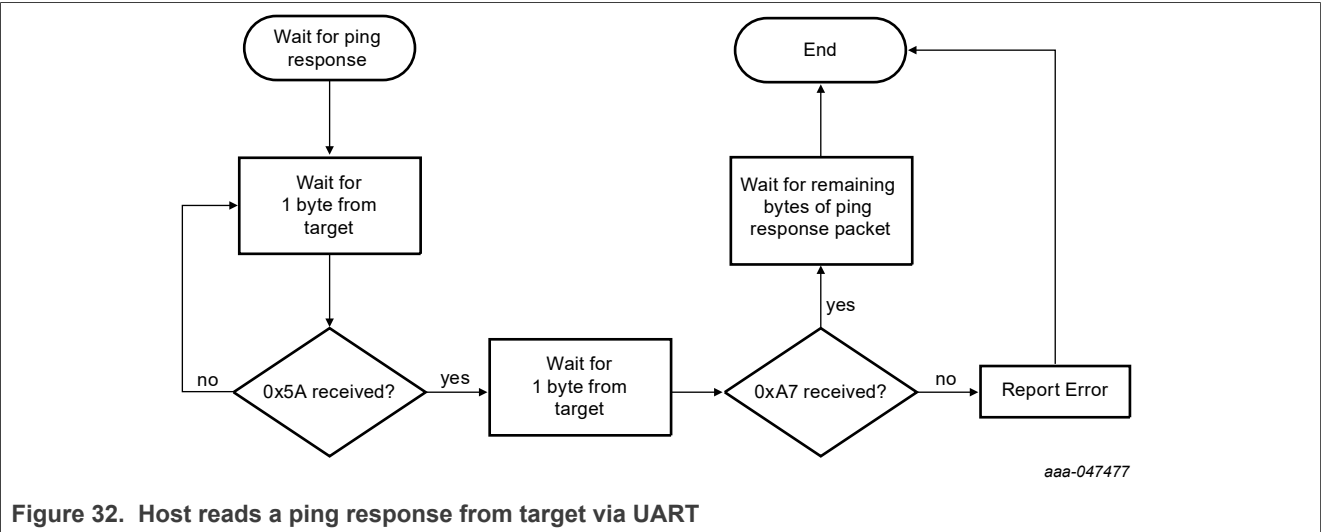


Figure 32. Host reads a ping response from target via UART

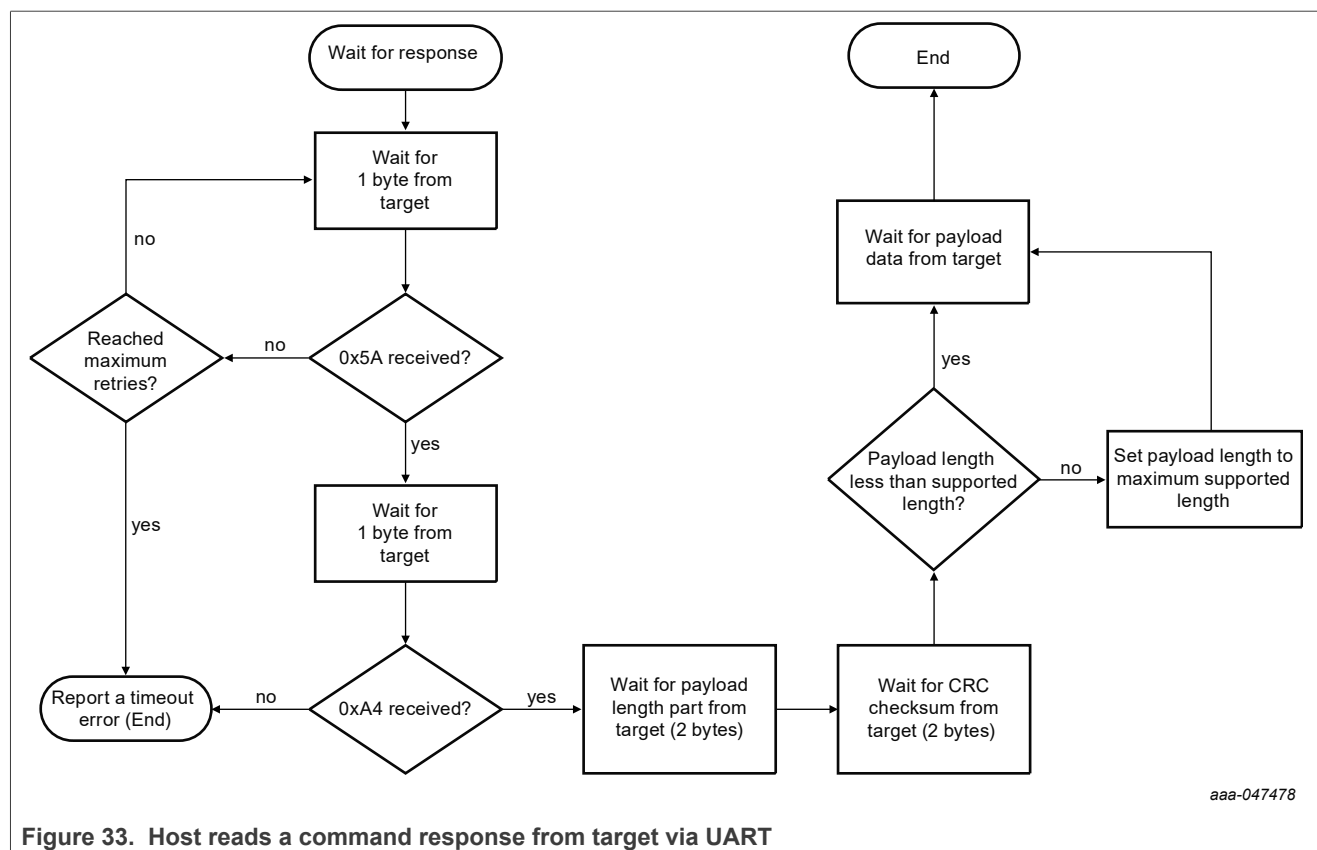


Figure 33. Host reads a command response from target via UART

8.6.2 UART ISP command format

For more details, see [Section 8.4](#).

8.6.3 UART ISP response format

For more details, see [Section 8.4](#).

8.6.4 UART ISP data format

For more details, see [Section 8.4](#).

8.6.5 UART ISP commands

For more details, see [Section 8.5](#).

8.7 I²C in-system programming

8.7.1 Introduction

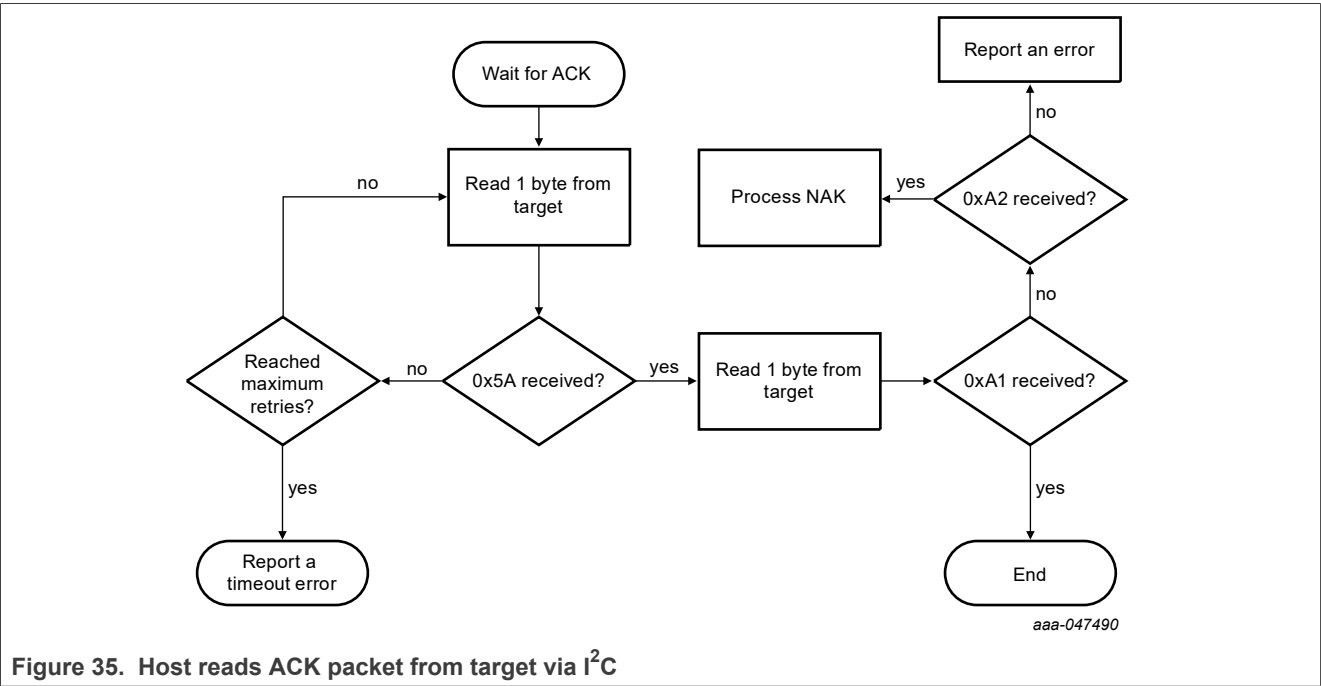
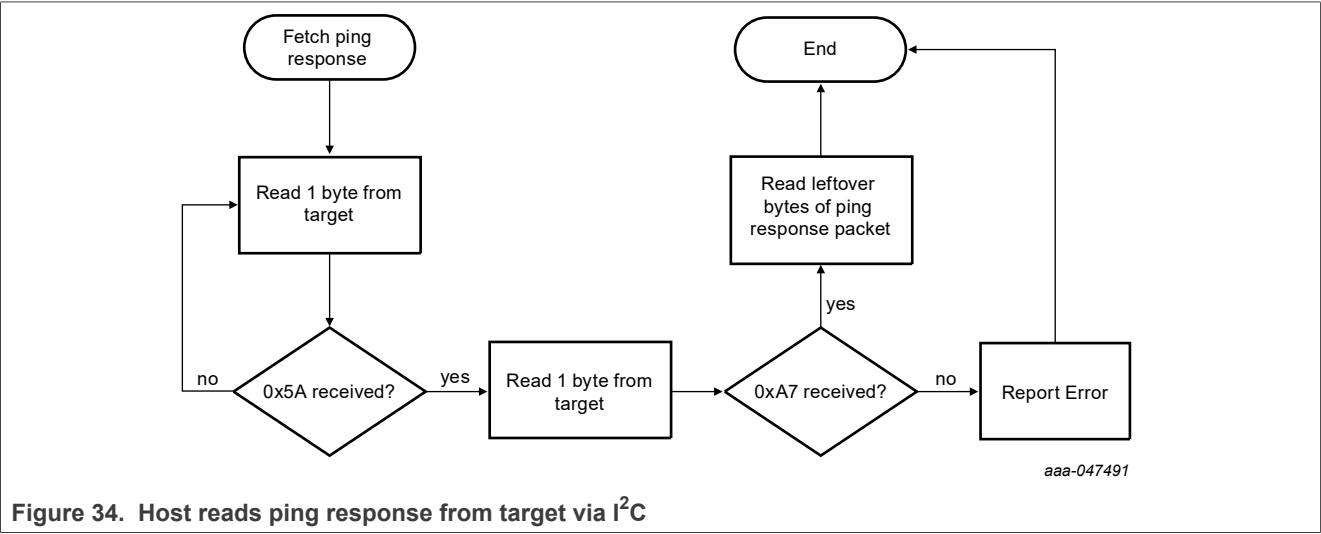
The bootloader supports loading data into flash via the I²C peripheral, where the I²C peripheral serves as the I²C slave. A 7-bit slave address is used during the transfer. The bootloader uses 0x10 as the I²C slave address and supports up to 400 kbit/s as the I²C baud rate.

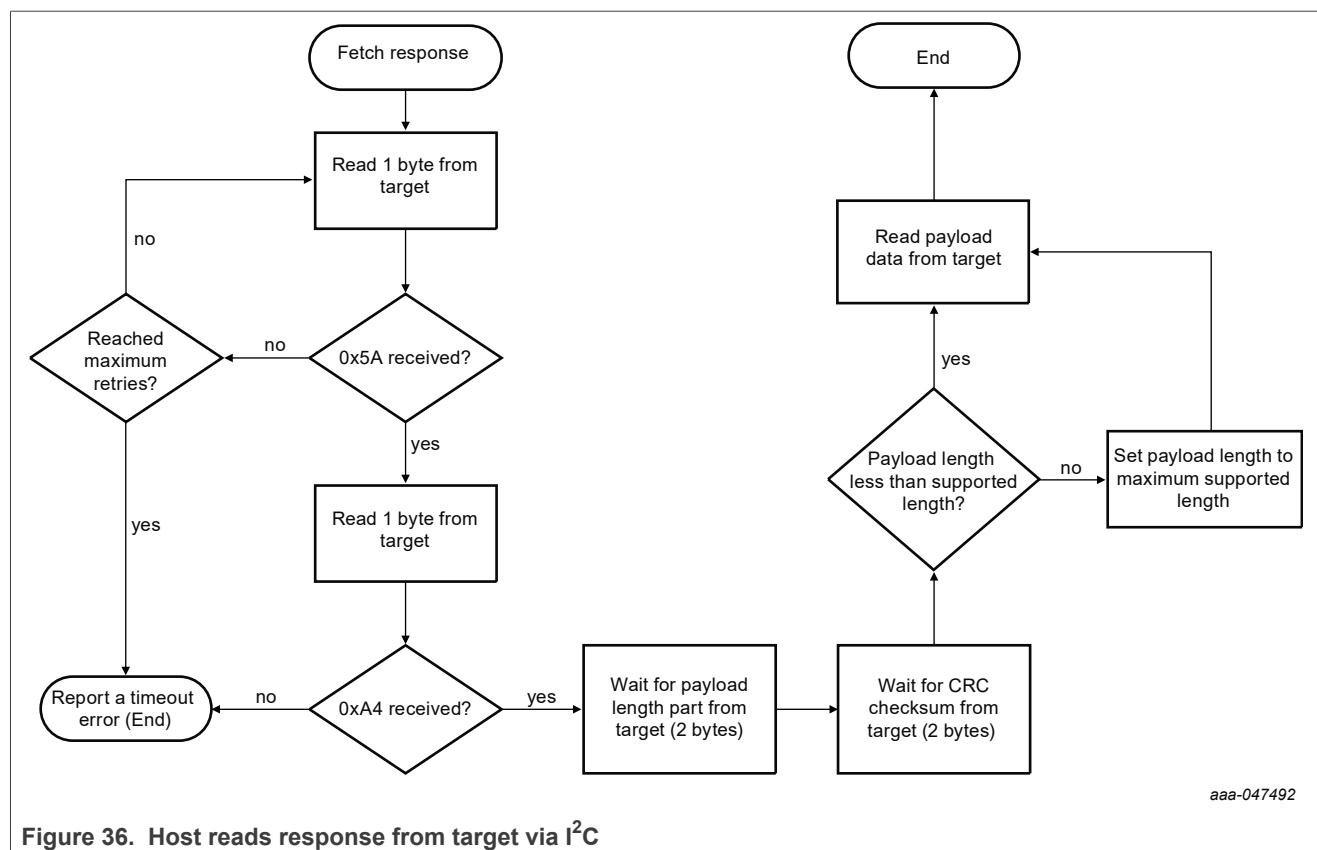
The maximum supported I²C baud rate depends on the core clock frequency when the bootloader is running. The typical baud rate is 400 kbit/s with factory settings.

Because the I²C peripheral serves as an I²C slave device, the host must start each transfer and fetch each outgoing packet.

- The host sends an incoming packet with a selected I²C slave address and the direction bit is set to write.
- The host reads an outgoing packet is read with a selected I²C slave address and the direction bit is set as read.
- If the target is busy with processing or preparing data, 0x00 is sent as the response to the host.

The following charts show the communication flow of the host reading the ping and ACK packets and the corresponding responses from the target.



Figure 36. Host reads response from target via I²C

8.7.2 I²C ISP command format

For more details, see [Section 8.4](#).

8.7.3 I²C ISP response format

For more details, see [Section 8.4](#).

8.7.4 I²C ISP data format

For more details, see [Section 8.4](#).

8.7.5 I²C ISP commands

For more details, see [Section 8.5](#).

8.8 SPI in-system programming

8.8.1 Introduction

The bootloader supports loading data into flash via the SPI peripheral, where the SPI peripheral serves as an SPI slave. The SPI transfer must be SPI Mode 3 with 8 data bits.

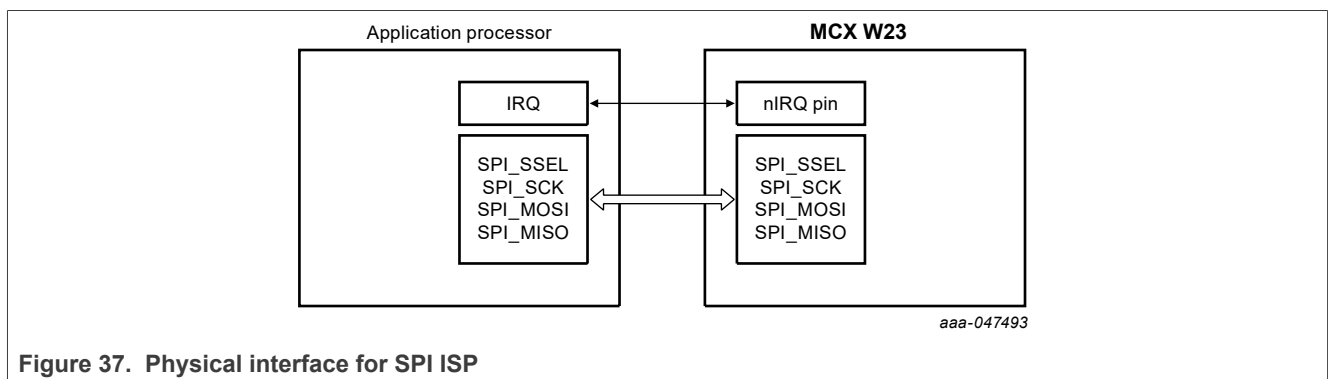
The maximum supported baud rate of the SPI depends on the core clock frequency when the bootloader is running. The typical maximum supported baud rate is 2000 kbit/s with the factory settings. The actual maximum supported baud rate can be lower or higher than 2000 kbit/s, depending on the actual value of the core clock.

Because the SPI peripheral serves as an SPI slave device, the host must start each transfer and fetch each outgoing packet.

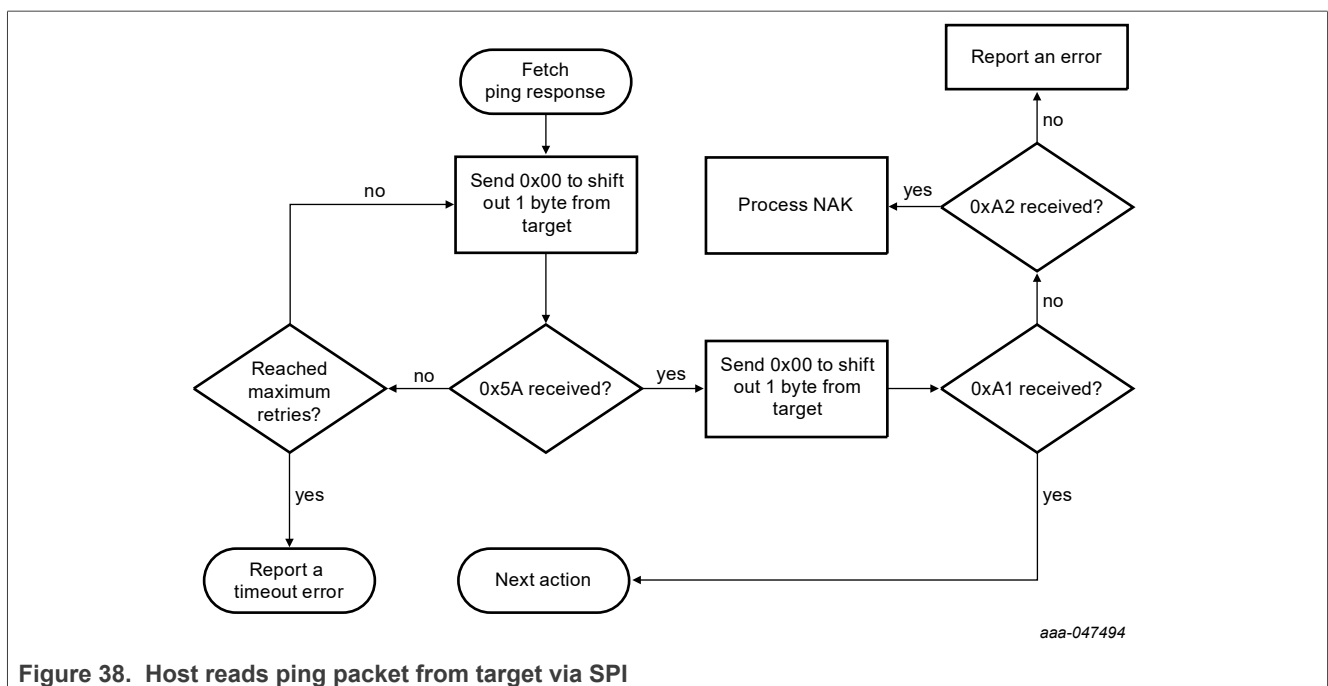
The transfer on SPI is slightly different from I²C:

- The host receives 1 byte after it sends out any byte.
- When the host is sending out bytes to the target, received bytes must be ignored
- The host starts reading bytes by sending 0x00s to target
If the target is under the following conditions, the byte 0x00 is sent as a response to host:
 - Processing incoming packet
 - Preparing outgoing data
 - Received invalid data

To notify the host processor that it is busy or ready for new commands/data, the bootloader also supports the active notification pin (nIRQ pin). For the typical physical connection between the host and the bootloader device, see [Figure 37](#).



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.



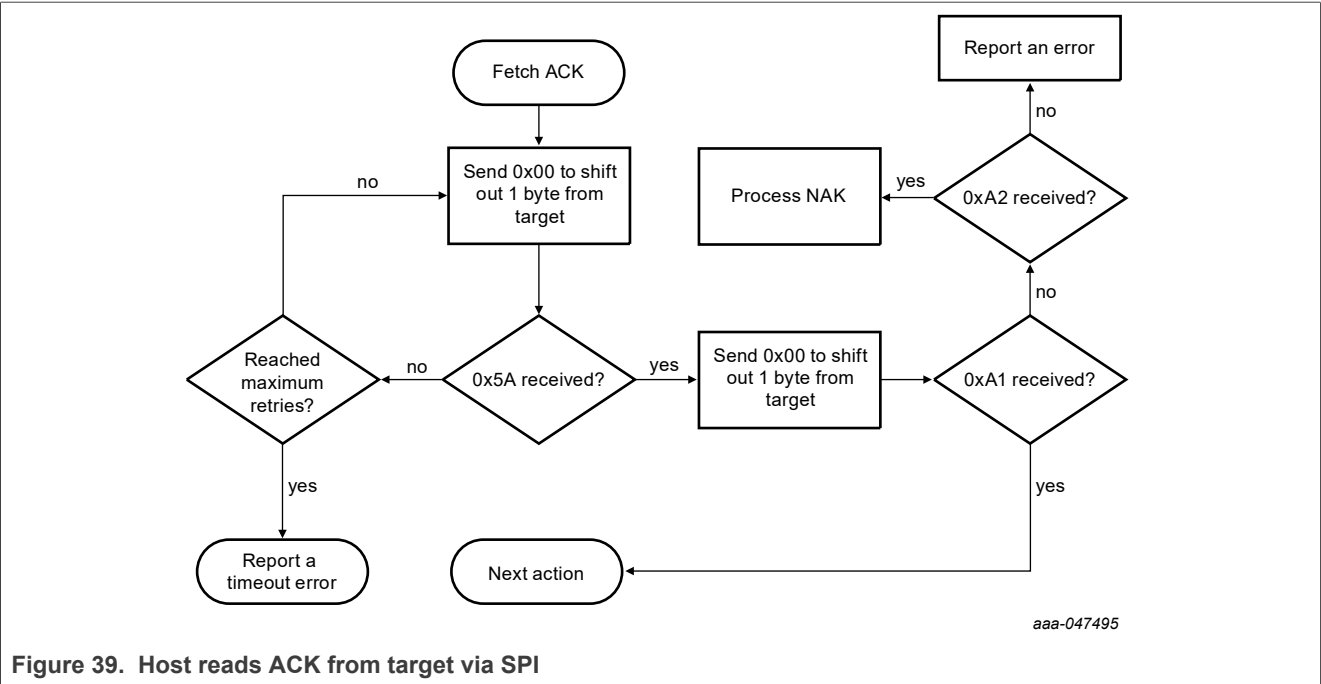


Figure 39. Host reads ACK from target via SPI

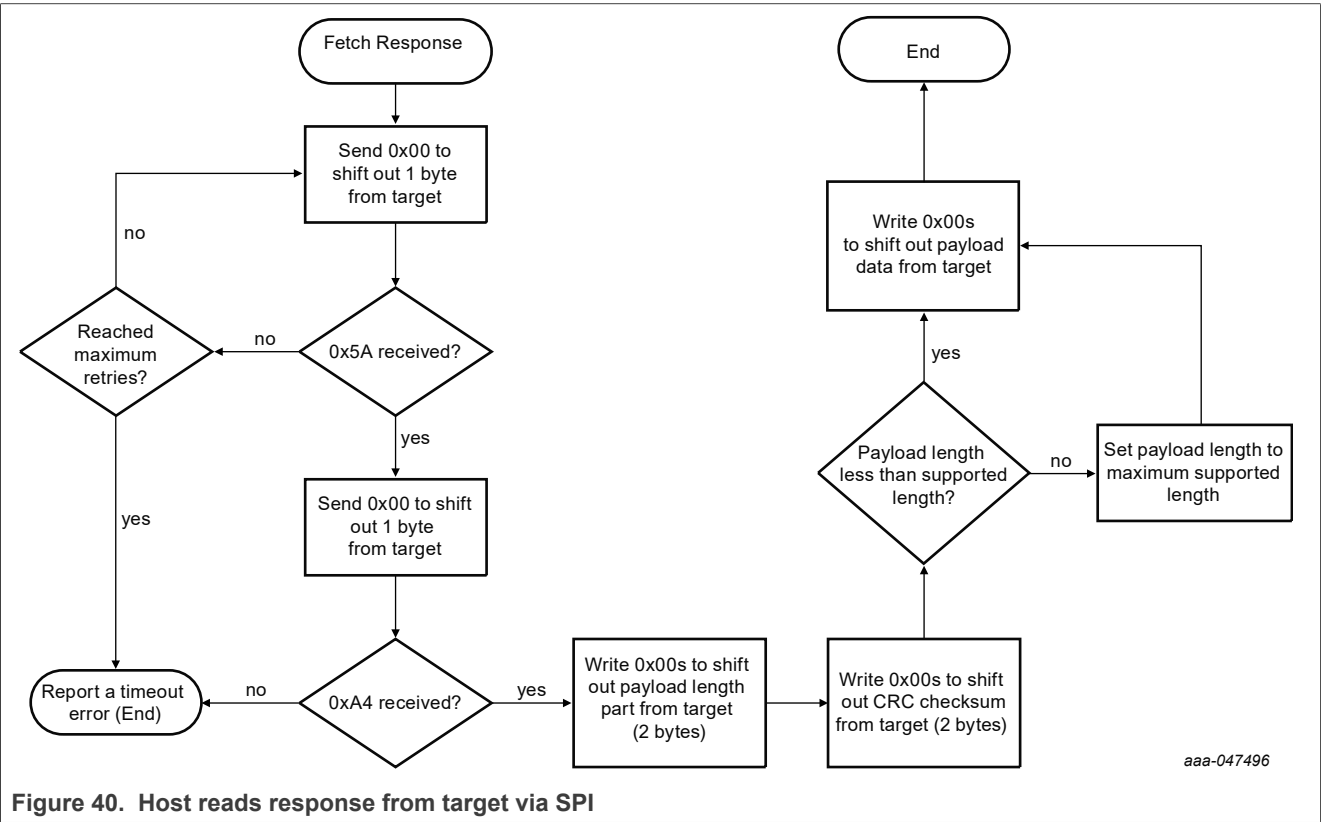


Figure 40. Host reads response from target via SPI

To accelerate the SPI transfer between the host and the bootloader, the bootloader provides an active notification pin known as the nIRQ pin, which the SetProperty command can enable. When enabled, the host

must wait until it sees a negative edge on the nIRQ pin before reading any data from the bootloader. It must also wait until the nIRQ pin is high before sending any data to the bootloader.

Note: *The SPI master must read back each byte in an ACK/response from the slave in 20 ms, otherwise, the ROM aborts the transfer.*

8.8.2 SPI ISP command format

For more details, see [Section 8.4](#).

8.8.3 SPI ISP response format

For more details, see [Section 8.4](#).

8.8.4 SPI ISP data format

For more details, see [Section 8.4](#).

8.8.5 SPI ISP commands

For more details, see [Section 8.5](#).

9 Flash API

This chapter applies to all MCX W23 parts.

9.1 Features

- The internal flash stores the following information:
 - The user application and the application data (in normal flash region)
 - The security life-cycle related parameter update (in PFR region)
- Boot ROM API support for programming the flash region and the protected flash region (PFR) region.

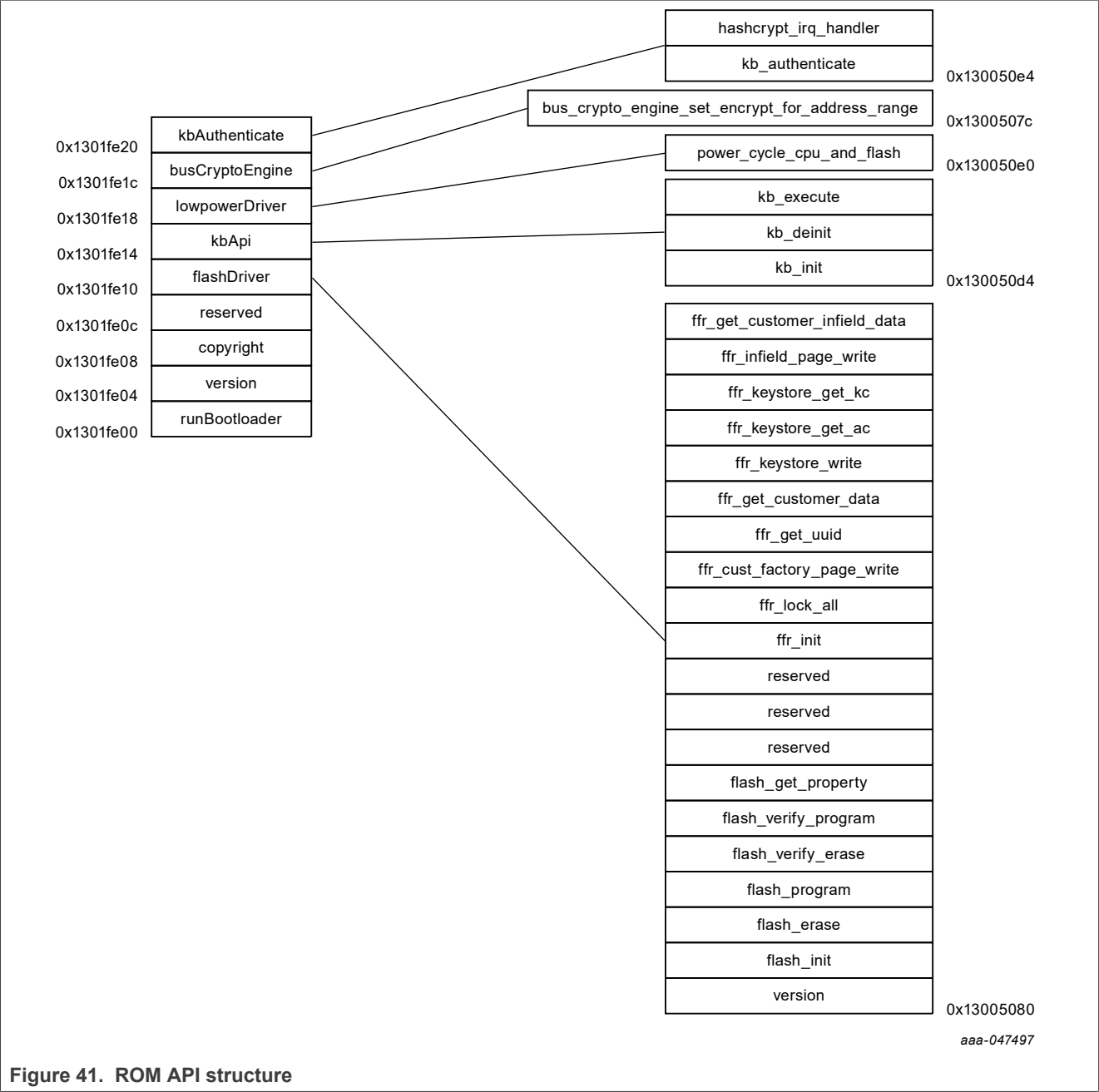
9.2 General description

This section describes the behavior of each flash driver API provided in the flash driver API tree.

To simplify the use of these flash driver APIs exported from the bootloader ROM, the `fsl_rom_api` driver is provided in the SDK. The `flash_iap` project in the `driver_examples` folder in the SDK shows how to use the flash API.

9.2.1 ROM API structure

The ROM API table is located at address `0x1301fe00` and contains pointers to the different exposed ROM APIs. The ROM APIs contain absolute ROM API function addresses which can be called using function pointers. The flash driver ROM API section starts at address `0x13005080` (see [Figure 41](#) for the ROM API layout).



The main purpose of the flash driver API is to provide access to functions used and implemented in ROM to do flash operations. The ROM API functions related to security are described in [Section 7.6](#).

Note: Some functions permanently return an error in some lifecycle states (see [Section 10.6](#) for more details) due to the restrictions the lifecycle state imposes.

Table 109. Flash ROM API summary

Address in ROM API table	Absolute function address	Function
0x13005084	0x13009A75	status_t flash_init(flash_config_t *config)
0x13005088	0x13009AAF	status_t flash_erase(flash_config_t *config, uint32_t start, uint32_t lengthInBytes, uint32_t key)
0x1300508C	0x13009AD9	status_t flash_program(flash_config_t *config, uint32_t start, uint8_t *src, uint32_t lengthInBytes)
0x13005090	0x13009B39	status_t flash_verify_erase(flash_config_t *config, uint32_t start, uint32_t lengthInBytes)
0x13005094	0x13000C3D	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)
0x13005098	0x13009C9D	status_t flash_get_property(flash_config_t *config, flash_property_tag_t whichProperty, uint32_t *value)
0x130050A8	0x130080B1	status_t ffr_init(flash_config_t *config)
0x130050AC	0x130080EB	status_t ffr_lock_all(flash_config_t *config)
0x130050B0	0x13008557	status_t ffr_cust_factory_page_write(flash_config_t *config, uint8_t *page_data, bool seal_part)
0x130050B4	0x13008885	status_t ffr_get_uuid(flash_config_t *config, uint8_t *uuid)
0x130050B8	0x1300861F	status_t ffr_get_customer_data(flash_config_t *config, uint8_t *pData, uint32_t offset, uint32_t len)
0x130050BC	0x1300863B	status_t ffr_keystore_write(flash_config_t *config, ffr_key_store_t *pKeyStore)
0x130050C0	0x130086CB	status_t ffr_keystore_get_ac(flash_config_t *config, uint8_t *pActivationCode)
0x130050C4	0x130086E9	status_t ffr_keystore_get_kc(flash_config_t *config, uint8_t *pKeyCode, ffr_key_type_t keyIndex)
0x130050C8	0x130082BB	status_t ffr_infield_page_write(flash_config_t *config, uint8_t *page_data, uint32_t valid_len)
0x130050CC	0x1300839B	status_t ffr_get_customer_infield_data(flash_config_t *config, uint8_t *pData, uint32_t offset, uint32_t len)

9.2.2 Flash APIs

This section describes each function supported in the flash driver API. The first word in the flash driver interface structure is the flash driver version.

The user of the flash driver allocates an instance of the flash configuration structure. It must be initialized by passing it to the flash_init() function first and then to the ffr_init() function. It is then passed into each of the functions as a parameter. The ROM code will fill in all members except sysFreqInMhz.

```
typedef struct _flash_config
{
    uint32_t sysFreqInMHz;           /* System frequency in MHz */
    flash_mem_desc_t flashDesc;      /* Flash description */
    flash_ffr_config_t ffrConfig;    /* Protected flash region description */
} flash_config_t;
```

The `flash_mem_desc_t` is defined as below:

```
typedef struct _flash_mem_descriptor
{
    uint32_t blockBase; /* Base address of the flash block */
    uint32_t totalSize; /* The size in bytes of the flash block */
    uint32_t blockCount; /* The number of flash blocks */
    uint32_t pageSize; /* The size in bytes of a page of flash */
    uint32_t sectorSize; /* The size in bytes of a sector of flash */
} flash_mem_desc_t;
```

The `flash_ffr_config_t` is defined as below:

```
typedef struct _flash_ffr_config
{
    uint32_t ffrBlockBase; /* Base address of the protected flash region
    block */
    uint32_t ffrTotalSize; /* The size in bytes of the protected flash
    region block */
    uint32_t ffrPageSize; /* The size in bytes of a protected flash region
    page*/
    uint32_t cfpaPageVersion; /* Version of the currently active CFPA page */
    uint32_t cfpaPageAddress; /* Base address of the currently active CFPA
    page */
} flash_ffr_config_t;
```

9.2.2.1 flash_init

This API is used to initialize 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 110. Parameters

Parameter	Description
config	Pointer to <code>flash_config_t</code> data structure in memory to store driver runtime state.

See the possible status codes in [Section 9.3](#).

9.2.2.2 flash_erase

This API is used to erase a specified flash range.

Note: No interrupts must be executed from flash while executing this function.

Prototype

```
status_t flash_erase(flash_config_t *config, uint32_t start, uint32_t
lengthInBytes, uint32_t key);
```

Table 111. 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 sector-aligned (that is, a multiple of 8192).
lengthInBytes	The length, given in bytes (not words or long words) to be erased. Must be sector-aligned.
key	Key is used to validate erase operation. Must be set to "kFLASH_ApiEraseKey"

See the possible status codes in [Section 9.3](#).

9.2.2.3 flash_program

This API is used to program data into specified flash region, the required start and the lengthInBytes must be phrase size aligned (16 bytes).

Note: No interrupts must be executed from flash while executing this function.

Prototype

```
status_t flash_program(flash_config_t *config, uint32_t start, uint32_t *src,
    uint32_t lengthInBytes);
```

Table 112. 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 phrase-aligned (16 bytes).
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 phrase-aligned.

See the possible status codes in [Section 9.3](#).

9.2.2.4 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.

Note: No interrupts must be executed from flash while executing this function.

Prototype

```
status_t flash_verify_erase(flash_config_t *config, uint32_t start, uint32_t
    lengthInBytes);
```


Table 113. 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 sector-aligned (8192 bytes).
lengthInBytes	The length, given in bytes (not words or long words) to be verified. Must be sector-aligned.

See the possible status codes in [Section 9.3](#).

9.2.2.5 flash_verify_program

This API is used to verify the data programmed in the flash memory. It compares the data with the expected data for a given flash area (as determined by the start address and length).

Note: No interrupts must be executed from flash while executing this function.

Prototype

```
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);
```

Table 114. 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	If not NULL, it is filled in with the failing address.
failedData	If not NULL, it is filled in with the failing data.

See the possible status codes in [Section 9.3](#).

9.2.2.6 flash_get_property

This API returns the requested flash property.

```
status_t flash_get_property(flash_config_t *config, flash_property_tag_t  
    whichProperty, uint32_t *value);
```

See [Table 116](#) for supported properties.

Table 115. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state
whichProperty	The requested property from the list of properties
value	Filled in with the value of the requested property

Table 116. Properties

Property	Value	Description
kFLASH_PropertyPflashSectorSize	0x00	Program flash sector size property
kFLASH_PropertyPflashTotalSize	0x01	Program flash total size property
kFLASH_PropertyPflashBlockSize	0x02	Program flash block size property
kFLASH_PropertyPflashBlockCount	0x03	Program flash block count property
kFLASH_PropertyPflashBlockBaseAddr	0x04	Program flash base address
kFLASH_PropertyPflashPageSize	0x30	Program flash page size property
kFLASH_PropertyPflashSystemFreq	0x31	System frequency in MHz
kFLASH_PropertyFfrSectorSize	0x40	PFR sector size property
kFLASH_PropertyFfrTotalSize	0x41	PFR total size property
kFLASH_PropertyFfrBlockBaseAddr	0x42	PFR base address property
kFLASH_PropertyFfrPageSize	0x43	PFR page size property

See the possible status codes in [Section 9.3](#).

9.2.3 PFR APIs

This section describes each function supported in the PFR driver API. The user of the PFR driver allocates an instance of the flash configuration structure. It must be initialized by passing it to the flash_init() function first and then to the ffr_init() function. For more information, see [Section 9.2.2](#).

Note: If a region has an SHA256 hash digest field programmed into the last 32 bytes (256 bits), FFR write and flash erase commands to FFR regions are inhibited.

9.2.3.1 ffr_init

This function is used to initialize the PFR controller and the flash_ffr_config context. It must be called before calling other PFR APIs.

Note: flash_init (see [Section 9.2.2.1](#)) must be called before calling the ffr_init API.

Prototype

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

Table 117. Parameters

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

See the possible status codes in [Section 9.3](#).

9.2.3.2 ffr_lock_all

This function is used to enable the firewall for all protected flash regions unconditionally. Writes are then inhibited unless a power-on reset (POR) or brownout detect (BOD) reset occurs.

Prototype

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

Table 118. Parameters

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

See the possible status codes in [Section 9.3](#).

9.2.3.3 ffr_cust_factory_page_write

This function is used to write a new CMPA page. It erases the existing CMPA page (see [Section 10.1](#) for details) and program the page with passed data.

Note: No interrupts must be executed from flash while executing this function.

Prototype

```
status_t ffr_cust_factory_page_write(flash_config_t *config, uint8_t* page_data, bool seal_part);
```

Table 119. 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 the CMPA page.
seal_part	If seal_part is TRUE, the routine computes an 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 must be set to TRUE to seal the part from additional modifications.

See the possible status codes in [Section 9.3](#).

9.2.3.4 ffr_get_uuid

This function is used to read the UUID from the NMPA.

Prototype

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

Table 120. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
uuid	Pointer to the destination buffer where the UUID will be copied to, the destination buffer must be able to hold a minimum of 16 bytes.

See the possible status codes in [Section 9.3](#)

9.2.3.5 ffr_get_customer_data

This function is used to read data stored in the CMPA (see [Section 10.1](#) for details).

Prototype

```
status_t ffr_get_customer_data(flash_config_t *config, uint8_t* pData, uint32_t offset, uint32_t len);
```

Table 121. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pData	Pointer to the destination buffer the data read from the CMPA page is copied to.
offset	Offset in the page to read data from.
len	Length in bytes of the data to read.

See the possible status codes in [Section 9.3](#).

9.2.3.6 ffr_keystore_write

This function is used to write a key store prepared in the RAM of the protected flash region for the key store. This action can only be done when the CMPA is not sealed (that is, the CMPA digest is completely 0).

Using this function must be avoided because no sanity checks are available on the provided key store in RAM. To set keys, use the ISP keyprovisioning command (see [Section 8.5.13](#)) instead.

Note: No interrupts must be executed from flash while executing this function.

Prototype

```
status_t ffr_keystore_write(flash_config_t *config, ffr_key_store_t* pKeyStore);
```

Table 122. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pKeyStore	Pointer to a key store prepared in RAM.

See the possible status codes in [Section 9.3](#).

9.2.3.7 ffr_keystore_get_ac

This API is used to get the activation code from the key store area. The calling code must pass a buffer pointer which can hold activation code (1192 bytes).

Note: Check if the flash aperture is small or regular and reads the data appropriately.

Prototype

```
status_t ffr_keystore_get_ac(flash_config_t *config, uint8_t* pActivationCode);
```

Table 123. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pActivation Code	Pointer to the destination buffer the activation code read from the key area is copied to. The destination buffer must be able to hold a minimum of 1192 bytes.

See the possible status codes in [Section 9.3](#).

9.2.3.8 ffr_keystore_get_kc

This function is used to get key codes from the key store area. The calling code must pass a pointer to a buffer which can hold a key code (52 bytes).

Prototype

```
status_t ffr_keystore_get_kc(flash_config_t *config, uint8_t* pKeyCode, ffr_key_type_t keyIndex);
```

Table 124. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pKeyCode	Pointer to the destination buffer the key code read from the key area is copied to; the destination buffer must be able to hold a minimum of 52 bytes.
keyIndex	See Table 125 for a list of allowed values.

The keyIndex must be one of:

Table 125. ffr_key_type_t

keyIndex	Value
kFFR_KeyTypeSbkek	0x105014
kFFR_KeyTypeUser	0x10504c
kFFR_KeyTypeUds	0x107d94
kFFR_KeyTypePrinceRegion0	0x107cb4
kFFR_KeyTypePrinceRegion1	0x107cec
kFFR_KeyTypePrinceRegion2	0x107d24
kFFR_KeyTypePrinceRegion3	0x107d5c
kFFR_KeyTypeAes	0x107dcc

See the possible status codes in [Section 9.3](#).

9.2.3.9 ffr_infield_page_write

This function is used to program the CFPA scratch page (see [Section 10.3](#) for details). The new page is copied to the CFPA ping or pong page (depending on which page is older). It becomes the active page after a reboot.

Note: No interrupts must be executed from flash while executing this function.

Prototype

```
status_t ffr_infield_page_write(flash_config_t *config, uint8_t* page_data,  
                                uint32_t valid_len);
```

Table 126. 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 the CFPA scratch page.
valid_len	The length in bytes to be programmed, the length must equal the protected page size (512 bytes).

See the possible status codes in [Section 9.3](#).

9.2.3.10 ffr_get_customer_infield_data

This function is used to read data stored in the CFPA page (see [Section 10.3](#) for details).

Prototype

```
status_t ffr_get_customer_infield_data(flash_config_t *config, uint8_t* pData,  
                                        uint32_t offset, uint32_t len);
```

Table 127. Parameters

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pData	Pointer to the destination buffer data read from the CFPA page is copied to.
offset	Offset in the page to read data from.
len	Length in bytes of the data to read.

See the possible status codes in [Section 9.3](#).

9.2.4 runBootloader API

The ROM API provides a function for the user application to enter the ISP mode based on the designated ISP interface mode. If successful, this function never returns to the user application. It performs a reset to return to the bootloader. In case of invalid arguments, the function returns to the user application.

Prototype

```
void (*runBootloader)(void *arg);
```

Table 128. Parameters

Parameter	Description
arg	Pointer to a word (4 bytes) that is interpreted as described in Table 129 .

The arg parameter points to a word that is interpreted in the following way:

Table 129. arg interpretation

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 - do not use
		2 - UART
		3 - SPI
		4 - I2C
Reserved	[15:04]	Not used
Image Index	[03:00]	Not used

9.3 Driver status codes

Table 130. Flash and PFR driver status codes

Status	Code (decimal)	Description
kStatus_FLASH_Success	0	The flash operation is successful.
kStatus_Fail	1	An unspecified error occurred.
kStatus_FLASH_InvalidArgument	4	Invalid argument detected during executing a FLASH API.
kStatus_FLASH_SizeError	100	Invalid CFPA page size detected during executing a FLASH API.
kStatus_FLASH_AlignmentError	101	Alignment error detected during executing a FLASH API.
kStatus_FLASH_AddressError	102	Address error detected during executing a FLASH API.
kStatus_FLASH_AccessError	103	Access error detected during executing a FLASH API.
kStatus_FLASH_ProtectionViolation	104	Protection 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_CommandAborOption	121	Command abort error detected during executing a FLASH API.
kStatus_FLASH_CompareError	122	Compare error detected during executing flash_erase_verify or flash_program_verify API.
kStatus_FLASH_FfrBankIsLocked	129	The FFR bank region is locked.
kStatus_FLASH_OutOfDateCfpaPage	132	CFPA page version is out of date.
kStatus_FLASH_BlankIfrPageData	133	Blank page cannot be read.
kStatus_FLASH_SealedFfrRegion	137	The FFR region is sealed.
kStatus_FLASH_FfrRegionWriteBroken	138	The FFR 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_CfpaScratchPageInvalid	148	The digest of the CFPA scratch page is incorrect.
kStatus_FLASH_CfpaVersionRollbackDisallowed	149	Rollback of certain features in the CFPA page is not allowed.

10 Protected flash region

The protected flash region is included on all MCX W23 devices.

The MCX W23 contains an internal protected flash region (PFR) which can be accessed using ROM APIs. During boot time, the ROM locks the PFR (based on the presence of the digest), which means that it cannot be overwritten.

Multiple regions are defined within the protected flash region. An overview of the regions can be found in [Section 7.2.1](#). Each of the regions is described in more detail in the following sections.

Full details of all fields can be found in the Register description ([Ref. 5](#)).

10.1 Customer manufacturing programmable area (CMPA)

The customer can program the CMPA during manufacturing of their product. There are two ways of programming the CMPA:

- Using the WriteMemory ISP command as described in [Section 8.5.7](#).
The CMPA does not have to be erased before writing to it, as the ROM handles writes to this region in a special way. The WriteMemory command must write the entire 512 bytes of the CMPA at once.
- Using the ROM API from an application as described in [Section 9.2.3.3](#).

Note: If the CMPA digest that is written is not entirely filled with 0, the ROM computes the digest and programs the computed digest as part of the CMPA. This action permanently seals the CMPA and the customer key store, making it impossible to change them again later. For other implications of writing the CMPA digest, see [Section 10.5](#).

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

- Boot configuration
Defines the boot speed where the core clock can be set to 12 MHz FRO or 32 MHz FRO. Default ISP mode (Auto ISP, UART ISP, SPI Slave ISP, I2C slave ISP, and Disable ISP). Boot failure indication using GPIO port pin.
- SPI flash configuration
Defines if recovery boot from 1-bit SPI flash is enabled. Recovery boot is described in [Section 6.3.3](#).
- Security policy for debug access control
DCFG_CC_SOCU_PIN and DCFG_CC_SOCU_DFLT
With TZ-M, level-1 customers (secure code developer) can sell the part to level-2 customers who develop non-secure code only. In this scenario, for ease of development, the level-1 customer releases the part to allow non-secure debug at all times. To allow level-2 customers to seal the part further, the DCFG_CC_SOCU_NS_PIN and DCFG_CC_SOCU_NS_DFLT words in CFPA are used. ROM uses these fields to restrict the debug access further.
DAP_VENDOR_USAGE_FIXED
Vendor usage field: The vendor uses it and its interpretation is application-specific. It can be used during debug authentication. 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. Newer debug credentials are required for debugging newer devices. If the vendor subdivides the VU attribute bitwise, a combination of these different interpretations can be supported.
- Secure boot flags (SECURE_BOOT_CFG) allow the user to:
 - Perform plain image boot with or without CRC
 - Enforce RSA-signed image boot
 - Enable or disable PUF key code generation

- Enable or disable DICE, and configure its CDI computation
- Enable or disable BOOT_SEED computation, and configure its computation
- Choose whether the use of TrustZone is mandatory or not
- Enforce the use of RSA-4096 keys
- PRINCE region configurations
- Root of trust keys table hash (RKTH)
- Flash remap configuration (FLASH_REMAP_SIZE and FLASH_REMAP_OFFSET)
Defines the location and size of a redundant boot image. If no valid image is present at address 0 of the internal flash, this image is booted.
- Bluetooth Unique Address
The MCX W23 devices come with a default Bluetooth Address stored in the NMPA. If the customer wants to use another Bluetooth Address for its devices, it can program these fields in the CMPA with the new Bluetooth Address to be used. The field BLE_UNIQUE_ADDRESS_VALID must be programmed with the value 0xC3A5 to allow the API to return the value in the BLE_UNIQUE_ADDRESS fields as Bluetooth Address.

Secure-boot-related features of the CMPA are described in [Section 7.2.2.1](#).

10.2 Customer key store

The customer can program the customer key store during manufacturing of their product. There are two ways of programming the customer key store:

- Using the KeyProvisioning ISP command as described in [Section 8.5.13](#).
- Using the ROM API from an application as described in [Section 9.2.3.6](#). This method must be avoided because there are no sanity checks on the provided key store.

Note: The customer key store can only be programmed when the CMPA digest is entirely filled with 0. When the CMPA digest is written with a non-zero value, the ROM seals the CMPA and the customer key store permanently, making it impossible to change them again later. For other implications of writing the CMPA digest, see [Section 10.5](#).

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

- Secure binary key encryption key (SBKEK key code) used for SB2 firmware update image decryption
- User key encryption key (USERKEK key code) used for user as preshared master key (256-bit symmetric key)

For more details, see [Section 7.2.3](#).

10.3 Customer field programmable area (CFPA)

The customer can program the CFPA at all times. However, restrictions on how values can be changed apply.

Three pages are used for the CFPA, two pages are in the independent protected area (ping and pong page) and one page that can be updated at the end of main flash area (scratch page). The bootloader code in ROM uses the ping or pong page with the higher version number as the active page.

There are two ways of programming the CFPA scratch page:

- Using the WriteMemory ISP command as described in [Section 8.5.7](#).
The CFPA scratch page does not have to be erased before writing to it, as the ROM handles writes to this region in a special way. The WriteMemory command must write the entire 512 bytes of the CFPA scratch page at once.
- Using the ROM API from an application as described in [Section 9.2.3.9](#).
A CFPA scratch page is only written if its version is higher than the version of the currently active page (ping or pong, depending on which has the highest version).

The update of the CFPA is not active immediately. Pages in the CFPA are updated using the following steps:

1. The scratch page which remains outside the protected region is updated.
2. Chip is reset to make the page effective.
3. On the subsequent boot, the ROM checks that the scratch page adheres to the restrictions described below.
4. ROM erases the oldest of the two protected pages (ping or pong page).
5. ROM copies the scratch page contents to the erased area.

An updated CFPA scratch page is only used when it adheres to the following restrictions:

- Several fields can only be higher than (increment only) or equal to the corresponding fields in the CFPA ping and pong pages:
 - Secure firmware version, used during SB2 file loading
 - Non-secure firmware version, used during SB2 file loading
 - Image key revocation version, which is checked during image authentication process
 - RoT key revocation for each of the four RoT keys
 - Vendor usage field that controls the allowed debug certificate version
The vendor usage 32-bit word is composed of two 16-bits halfwords that are complementing each other. Since only a higher word can update this 32-bit word, it is advised to start with 0x0000FFFF. It allows using 0x0001FFFE as a next value because it is higher, which means that the effectively used 16-bit word decreases from 0xFFFF to 0xFFFE.
- **Note:** On A0 silicon, the entire 32-bit word is used for rollback prevention. Only a higher word can update it. So, it is advised to start with 0x0000FFFF. Starting with 0x0000FFFF allows using 0x0001FFFE as a next value because it is higher. It means that the effectively used 16-bit word decreases from 0xFFFF to 0xFFFE.
- Enable FA mode (which erases all secrets and prepares the chip for failure analysis); when it is enabled, it cannot be disabled.

In the CFPA, the user can set the following features:

- The fields with restrictions that were already mentioned above.
- Space is reserved for additional customer-specific settings that have no influence on the ROM behavior.

Secure boot-related features of the CFPA are described in [Section 7.2.2.2](#).

10.4 NXP programmed area

In this region, the user can find the UUID (universal unique identification ID), which can be used to identify a device.

10.5 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 lifecycle state to cross check integrity of the region.

Programming the CMPA hash digest has several effects:

- It permanently disables erase/write commands to the CMPA and Customer Key Store regions by the ROM or by flash API commands executed by the run-time code.
- It moves the device out of the Tier 1 development lifecycle state. This permanently disables the FlashEraseAll, FlashEraseRegion, ReadMemory, WriteMemory, FillMemory, ConfigureMemory and Execute ISP commands. GetProperty, Reset, KeyProvisioning, SetProperty, and ReceiveSbFile are still allowed.
- The Debug Authentication configuration written to the CMPA cannot be relaxed anymore, only made more strict by writing to the Debug Authentication configuration in the CFPA.

10.6 Customer development lifecycle state

Initially, customers use the MCX W23 device in the "NXP provisioned" lifecycle state. Depending on application security targets, customers can ship the MCX W23 in this "NXP provisioned" state or move it to the "OEM provisioned" state, where the user-programmable domain requires an authenticated image to boot.

The "Failure Analysis" state is used for retiring the MCX W23. The ROM automatically erases various secrets that NXP and the Tier1/2/OEM provided. This state allows FA testing to be run on customer returns. The PUF output is blocked completely and the part does not boot any images, it only provides dedicated access for failure analysis.

The details on each of the states (for example, PFR fields that must be programmed and their impact on each subsystem) and transitioning between states (PFR fields that must be programmed) are provided in [Table 131](#).

Note: Full details of all PFR fields can be found in the Register description ([Ref. 5](#)).

Table 131. Lifecycle state descriptions

Lifecycle state	Descriptions	PFR fields to provision	Transitions		
			Fields to write	Next state	Notes
NXP provisioned	State as it left the NXP factory towards an OEM	N/A	N/A	OEM Open	This change in lifecycle is more of a location change, which happens when the device gets shipped from an NXP Facility to the OEM Factory
			N/A	Bricked	-

Table 131. Lifecycle state descriptions...continued

Lifecycle state	Descriptions	PFR fields to provision	Transitions	Next state	Notes
			Fields to write		
OEM Open / OEM Tier 1	Tier 1 software development state: - ROM opens debug ports. However debug access is disabled during ROM execution. - If debug authentication fields (DCFG_CC_SOCU_xxx) are programmed, they are used to determine debug access. - SECURE_BOOT_CFG field determines if secure boot flow is enabled or not, as well as other options for TrustZone, and boot seed.	CMPA fields are programmable: Secure boot: - RKTH - BOOT_CFG - SECURE_BOOT_CFG PRINCE: - PRINCE_CFG - BMAPCTX0 - BMAPCTX1 - BMAPCTX2 - BMAPCTX3 Debug authentication: - DCFG_CC_SOCU_DFLT - DCFG_CC_SOCU_PIN - VENDOR_USAGE Customer key store: - SBKEK - USERKEK Optional: - Customer defined fields - CFPA fields	- CMPA_DIGEST is programmed. - DCFG_CC_SOCU_PIN and DCFG_CC_SOCU_DFLT should be set (each feature is either disabled permanently or accessible through debug authentication). See Section 43.6.2 for a detailed definition of these bits - Secure world debug should be restricted or disabled	OEM Tier 2	This state is intended for when the Tier 1 ships the device with a secure world application for a Tier 2 to write a non-secure world application
			- CMPA_DIGEST is programmed. - DCFG_CC_SOCU_PIN and DCFG_CC_SOCU_DFLT should be set (each feature is either disabled permanently or accessible through debug authentication). See Section 43.6.2 for a detailed definition of these bits	OEM provisioned	This state is intended for when the Tier 1 writes both TZ world applications or a non-TZ implementation
			N/A	Bricked	-

Table 131. Lifecycle state descriptions...continued

Lifecycle state	Descriptions	PFR fields to provision	Transitions	Next state	Notes
			Fields to write		
OEM Tier 2	Tier 2 software development state: - Secure debug ports closed or protected using debug authentication. - Non-secure debug ports open - Primary image (Tier 1) enables TZ-M. - CMPA cannot be written. - ISP commands are limited or disabled.	CFPA fields are programmable through secure API calls exposed by customer code. DCFG_CC_SOCU_NS_PIN and DCFG_CC_SOCU_NS_DFLT fields in CFPA are used to close the part further. Note: These fields can only be used if the DCFG_CC_SOCU_PIN and DCFG_CC_SOCU_DFLT have been programmed in CMPA before and will only strengthen the security level.	CFPA fields DCFG_CC_SOCU_NS_PIN and DCFG_CC_SOCU_NS_DFLT should be set (each feature is either disabled permanently or accessible through debug authentication) See Section 43.6.2 for a detailed definition of these bits	OEM provisioned	-
			N/A	Bricked	-
OEM Provisioned	In-field application state: - Debug ports closed or protected using debug authentication - TZ-M optional (depends on settings in the SECURE_BOOT_CFG field) - ISP commands are limited or disabled - CMPA cannot be written - CFPA can only be written using the scratch page mechanism	CFPA fields are programmable through secure API as required through the product life: - IMAGE_KEY_REVOKE - S_FW_VERSION - NS_FW_VERSION - VENDOR_USAGE (debug key revocation)	N/A	Application Active	This change in lifecycle is more of a location change, which happens when the device gets shipped from an OEM to their customer
			N/A	Bricked	-

Table 131. Lifecycle state descriptions...continued

Lifecycle state	Descriptions	PFR fields to provision	Transitions		
			Fields to write	Next state	Notes
Application Active	State as it left the OEM Factory towards the end-user	Same as "OEM Provisioned"	N/A	OEM Closed - No return	-
			CFPA field ENABLE_FA_MODE is set	Failure Analysis	The following sequence of operations is required before transitioning a part in "Failure Analysis": 1. Erase flash (mass erase, except PFR) 2. Erase system SRAM 3. Erase PUF key store 4. Set ENABLE_FA_MODE field 5. Trigger reset
			N/A	Bricked	-
OEM Closed - No Return	End-state at customer, device will not be in use, nor will it be returned. • Debug ports are disabled. • OEM assets removed	N/A	N/A	Bricked	-

Table 131. Lifecycle state descriptions...continued

Lifecycle state	Descriptions	PFR fields to provision	Transitions		
			Fields to write	Next state	Notes
Failure Analysis	Returned (CQI): • ROM checks that the key store is empty and blocks PUF key unwrapping functionality before enabling test/debug ports open. • ROM does not 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 and CFPA cannot be written	N/A	N/A	Bricked	-
Bricked	Fail/End-of-Life state: • Debug ports are disabled. • All assets are removed.	N/A	N/A	N/A	-

11 Analog control (ANACTRL)

The analog controller is available on all MCX W23 devices.

11.1 Features

- Internal free running oscillator (FRO). This oscillator provides a selectable 32 MHz output, a 24 MHz, and a 12 MHz output that can be used as a system clock. Additionally, there are a 64 MHz, a 48 MHz, and a 32 MHz output that can be used as a SPIFI clock.
- 32.768 kHz internal FRO.
- High-accuracy frequency measurement function for on-chip and off-chip clocks.
- All brownout detectors (BoD) and DCDC converter interrupts generation control and status.

11.2 Basic configuration

- To enable the clock to the analog controller module, set the ANALOG_CTRL bit in the AHBCLKCTRL2 register in the section [Section 4](#) .
Note: The clock to the analog controller module is enabled during boot time by the bootloader and must always remain enabled.
- Target and reference clocks for the frequency measurement function are selected in the input multiplexer block (see the sections [Section 45.4.5.2.9](#) and [Section 45.4.5.2.10](#)).
- The 32.768 kHz free running oscillator is enabled when:
 - The RTC_OSC_PD bit in RTC control register = 0 and the SEL bit in PMC.RTCOSC32K register = 0.
 - the OSC_PD bit in the PMC.OSTIMER register = 0 and the SEL bit in the PMC.RTCOSC32K = 0.
 - The PDEN_FRO32K bit in PDRUNCFG0 = 0.

11.2.1 Measure the frequency of a clock signal

The frequency of an 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 determine the frequency of the 32.768 kHz free running oscillator (FRO32K) accurately.

The clock frequency to be measured and the reference clock are selected in the input mux block (see the sections "Selection for frequency measurement reference clock (FREQMEAS_REF)" and "Selection for frequency measurement target clock (FREQMEAS_TARGET)" in the Register description ([Ref. 5](#))). Details on the accuracy and measurement process are described in [Section 11.4.1](#). To start a frequency measurement cycle and read the result, see [Section 11.4.1](#).

11.3 Pin description

The analog controller has no configurable pins.

11.4 Function description

11.4.1 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 42](#) shows a block diagram of the frequency measure function.

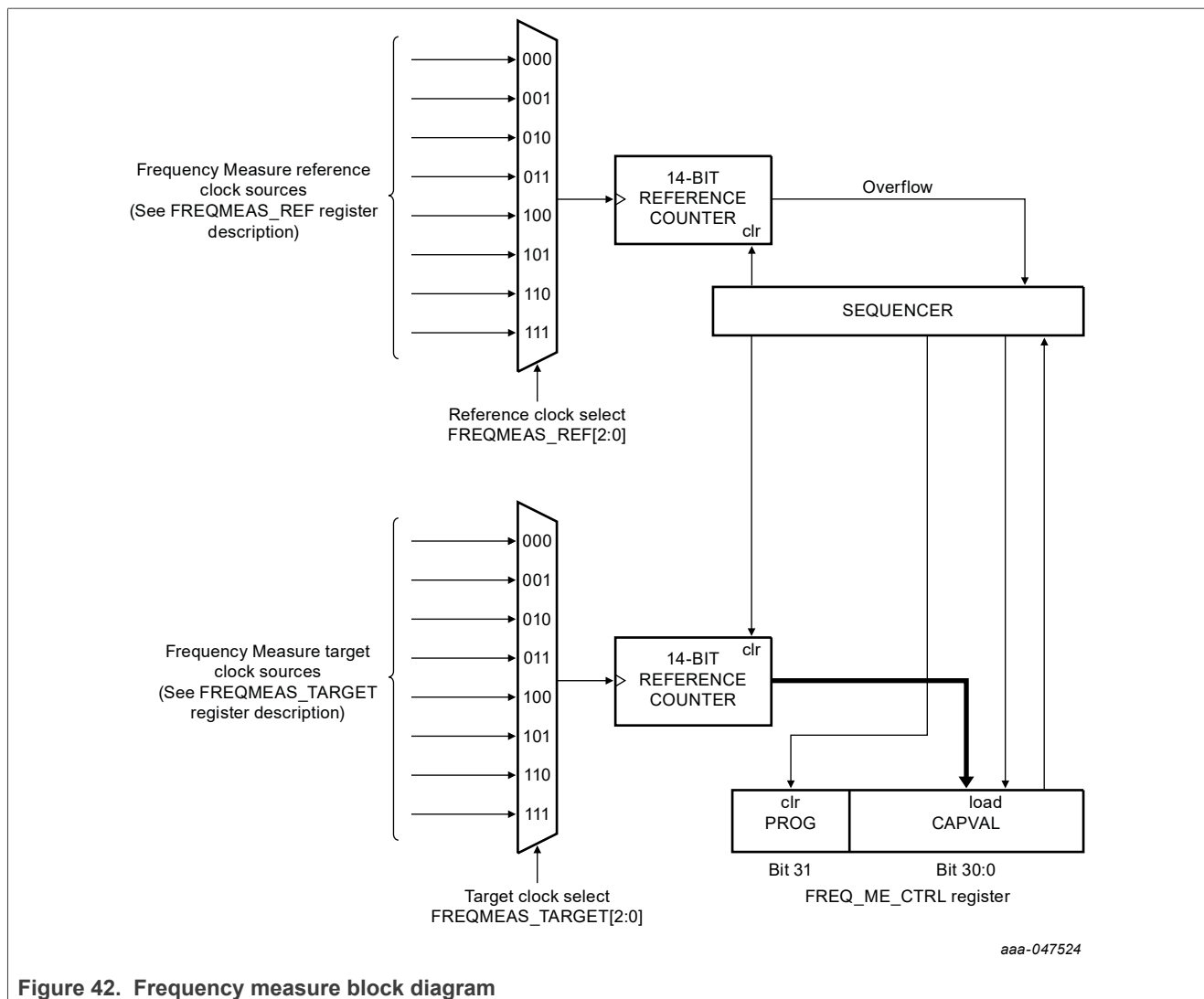


Figure 42. Frequency measure block diagram

The target and reference clocks are selectable by programming target clock select `FREQMEAS_TARGET` register and reference clock select `FREQMEAS_REF` register (see the section "Input multiplexing (INPUT MUX)" in the Register description ([Ref. 5](#)).

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.

Software initiates a measurement cycle, setting the 5 bits `SCALE` value `FREQ_ME_CTRL[4:0]` and setting the measurement-in-progress bit `PROG` in the `FREQ_ME_CTRL` register (see the section "Frequency measure function control register (`FREQ_ME_CTRL`)" in the Register description ([Ref. 5](#)).

The software can then poll this same measurement-in-progress `PROG` bit `FREQ_ME_CTRL[31]` which hardware clears when the measurement operation is completed.

When the reference counter rolls-over, after $(2^{\text{SCALE}} - 1)$ reference clock edges, the measurement cycle terminates. At that point, the state of the target counter is loaded into the capture field `CAPVAL` = `FREQ_ME_CTRL[30:0]` and the measure-in-progress bit `PROG` = `FREQ_ME_CTRL[31]` is cleared.

Software can read this capture value and apply a specific calculation to it, which returns the frequency of the target clock in MHz according to following formula:

Example: Reference clock is 1 MHz and target clock is 32 K OSC. SCALE = 11

Step 1: `FREQMEAS_TARGET[4:0] = 4; FREQMEAS_REF[4:0] = 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: $F_{\text{target}} = (\text{CAPVAL} * 1^6) / ((1 \ll 11) - 1)$

Note: Before performing this function, both clocks (reference and target) must be enabled. If there is a large difference in frequency between the two clocks, configure the clocks so that the slowest clock is input as reference clock. This configuration provides the highest level of accuracy.

11.4.1.1 Accuracy

The frequency measurement function can measure the frequency of an on-chip (or off-chip) clock (referred to as the target clock) to a high degree of accuracy by using another on-chip clock of known frequency as a reference.

The following constraints apply:

- The frequency of the reference clock must be (somewhat) greater than the frequency of the target clock.
- The system clock used to access the frequency measure function register must also be greater than the frequency of the target clock.

The frequency measurement function circuit can measure the target frequency with an error of less than 0.1 %, provided the reference frequency is known precisely. Uncertainty in the reference clock (for example the ± 1 % accuracy of the FRO) adds to the measurement error of the target clock. In general, though, its additional error is less than the uncertainty of the reference clock.

If the reference frequency exceeds the target frequency by a very large margin (25x or more), a modest loss of accuracy can occur. However, accuracy is not a simple function of the magnitude of the frequency difference. Nearly identical frequency combinations, still with a spread of about 43x, result in errors of less than 0.05 %. If the target and reference clocks are different by more than a factor of approximately 500, then the accuracy decreases to ± 4 %.

12 Crystal oscillator (XTAL32M - XTAL32K)

All MCX W23 products have two crystal oscillators, a 32 MHz crystal oscillator (also referred as high-speed crystal oscillator) and a 32.768 kHz crystal oscillator (also referred as low-speed crystal oscillator).

The low-speed crystal oscillator (XTAL32K) has one embedded capacitor bank. The capacitor banks can be used as 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 2 pF and 25 pF (IEC equivalent). So, they allow saving component area on the PCB and in the bill of materials (BOM).

The transconductance (gm) of the XTAL32K oscillator gain stage can be determined by the gm register field. The required gm depends on the crystal specifications.

In [Table 132](#) examples are given of capacitance value and gm setting for a certain crystal specification. Since the XTAL32K consist of an internal capacitorbank on the oscillator input pin XI_32K and on oscillator output pin XO32K, only a crystal needs to be connected externally. For low power consumption it is recommended to use a crystal with a CL=6pF and low ESR value in combination with the lowest amplitude setting ampl[1:0]=0. The oscillation amplitude can be increased up to setting ampl[1:0]=3. A larger oscillation amplitude setting will make the oscillator less sensitive for interference.

Table 132. Example of capacitance value and gm setting for a certain crystal specification

Crystal Frequency f_c [kHz]	Max ESR R_s [kOhm]	Max shunt capacitance C_p [pF]	Crystal load capacitance C_L [pF]	Capacitance value at pin XI_32K & XO_32K $C_{x1} = C_{x2}$ [pF]	gm[1:0] Mode
32.768	90	1.6	4	~8	0
	90	1.6	6	~12	1
	90	1.6	9	~18	2
	90	1.6	12.5	~25	3

The high-speed crystal oscillator (XTAL32M) does not contain an embedded capacitor bank. External load capacitors are required to tune the crystal frequency. The "Hardware design considerations for MCX W23" application note ([Ref. 9](#)) describes in more detail how to do the crystal tuning for the high-speed crystal oscillator.

12.1 Features

- Conserves component area on the PCB.
- Save on bill of materials (BOM).
- Relies on simple APIs to configure the capacitor banks based on the crystal capacitive load (CL) and measured PCB parasitic capacitances on the XI32K and XO32K pins.

12.2 Crystal oscillator API description

Table 133. XTAL32K API calls

Function prototype	API description	Section
void POWER_XTAL32K_ConfigureCaps(uint8_t capIn, uint32_t capOut);	Function to configure the caps of the 32.768 KHz crystal. The actual load capacitance is calculated as • XTAL32K Load Cap = 0.4 pF + 1.64 pF x capIn • XTAL32K Load Cap = 0.4 pF + 1.64 pF x capOut	Table 153

Table 133. XTAL32K API calls...continued

Function prototype	API description	Section
void POWER_XTAL32K_ConfigureReferenceOutputCurrent(uint8_t gm);	Function to configure the reference output current of the 32.768 KHz crystal.	Table 154
void POWER_XTAL32K_ConfigureOscillatorAmplitude(uint8_t ampl);	Function to configure the oscillator amplitude of the 32.768 KHz crystal	Table 155
void POWER_XTAL32K_EnableBypass(bool enable);	Function to configure 32.768 KHz crystal oscillator to bypass mode. Caps will be set to 0 and XO32K block will be powered when bypass is enabled.	Table 156
void POWER_XTAL32K_SetStartupTime(uint32_t startupTime);	Function to configure 32.768 KHz crystal oscillator start up time. This function overrides the default start up time of 500 ms.	Table 157

Table 134. XTAL32M API calls

Function prototype	API description	Section
void POWER_XTAL32M_BLE_GateEnable(bool enable);	Function to gate 32MHz BLE crystal	Table 158
status_t POWER_XTAL32M_SetStartupTime(uint8_t startupTime);	Function to configure 32MHz XTAL oscillator start up time. This function overrides the default start up time of 288 us. The resolution is 32 us. Only values from 1 until 15 are allowed (1=32us, 15=480us).	Table 159

13 Power management

This chapter provides an overview of power-related information about MCX W23 devices. These devices include various 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, the power API functions `POWER_PeripheralPowerOn` and `POWER_PeripheralPowerOff` can be used to configure the `PMC_PDRUNCFG0` register (see [Table 138](#) and the section "Power-down configuration during active mode (PDRUNCFG0)" in the Register description ([Ref. 5](#)).

In deep-sleep, power-down, and deep-power down modes, the power profile API controls which analog peripherals remain powered up (see [Section 13.2](#)).

13.1 General description

13.1.1 Power supplies

There are four supply pins. The selected application mode determines which pins must be supplied.

- **VBAT_LV**
Main supply if external regulated supply mode (XR_SM)
- **VBAT_HV**
Main supply if high-voltage supply mode (HV_SM) and external regulated supply mode (XR_SM)
- **VDD_IO**
Supply for the digital IO of the part
- **VDD_RF**
Supply for the radio subsystem of the part

To provide power to the different submodules, the MCX W23 contains multiple on-chip regulators.

- DCDC converter
- LDO_AO regulator (always-on power domain regulator)
- LDO_MEM regulator (SRAM regulator)
- LDO_DIG regulator (digital core regulator)
- LDO_XO_32M regulator (32 MHz crystal oscillator regulator)

13.1.2 Power domains

The device is partitioned into six power domains:

- **PD_CORE** (power domain core)
Most of all digital core logic (CPU0, multilayer matrix, and almost all peripherals like Flexcomm, SDMA).
- **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. As long as sufficient voltage is available on VBAT_HV or VBAT_LV and the device is not in POWEROFF mode, this domain always has power.
- **PD_MEM** (power domain memories)
All SRAM instances.
- **PD_RF** (power domain radio)
Radio subsystem.
- **PD_WUP** (power domain wake-up)
Wake-up controller.

[Table 135](#) shows the detailed list of all modules per power domain.

Table 135. Power domain supply

	PD_CORE	PD_SYSTEM	PD_AO	PD_MEM	PD_RF	PD_WUP
input/output	-	GPIOs other than the wake-up GPIO pin	-	-	-	Wake-up GPIO pin
analog components	192 MHz free running oscillator (FRO192M)	-	32.768 kHz crystal oscillator (XTAL32K)	-	32 MHz crystal oscillator	-
	-	-	32.768 kHz free running oscillator (FRO32K)	-	Bluetooth Low Energy PHY	-
	-	-	1 MHz free running oscillator (FRO1M)	-	-	-
	-	-	brownout detector 1	-	-	-
	-	-	brownout detector 2	-	-	-
	-	-	DCDC converter	-	-	-
digital components	Arm Cortex-M33	reset controller	power management controller (PMC)	-	Bluetooth Low Energy MAC	-
	debug ports (SWD interface)	system configuration (SYSCON)	real-time clock (RTC)	-	Bluetooth Low Energy Sleep Timer (BLE_SLP_TMR)	-
	mailbox	I/O configuration (IOCON)	OS event timer (OSTIMER)	-	-	-
	debug mailbox	sleep controller (SLEEPCON)	-	-	-	-
	CASPER	I/O functional multiplexers	-	-	-	-
digital components (cont'd)	PRINCE	group GPIO input interrupt (GINT0)	-	-	-	-
	system DMA	secret keys (extracted from PUF)	-	-	-	-
	secure system DMA	Flexcomm interface 0	-	-	-	-

Table 135. Power domain supply...continued

	PD_CORE	PD_SYSTEM	PD_AO	PD_MEM	PD_RF	PD_WUP
	general-purpose I/O controller	-	-	-	-	-
	secure general-purpose I/O controller	-	-	-	-	-
	Flexcomm interface 1 and 2	-	-	-	-	-
	hash/AES crypto engine	-	-	-	-	-
	CRC engine (CRC)	-	-	-	-	-
	SCTimer/PWM (SCT)	-	-	-	-	-
	standard counter/timer 0,1,2,3,4 (CTIMER)	-	-	-	-	-
	ROM/SRAM/flash controllers	-	-	-	-	-
	microtick timer	-	-	-	-	-
	multirate timer (MRT)	-	-	-	-	-
	windowed watchdog timer (WWDT)	-	-	-	-	-
	true random number generator (RNG)	-	-	-	-	-
	programmable logic (PLU/LUT)	-	-	-	-	-
	physically unclonable function (PUF)	-	-	-	-	-
digital components (cont'd)	analog controller (ANACTRL)	-	-	-	-	-
	peripheral input multiplexing	-	-	-	-	-
	pin interrupt and pattern matching (PINT)	-	-	-	-	-
memories	ROM (256 kB)	-	-	RAM_X0 (4 kB)	-	-

Table 135. Power domain supply...continued

	PD_CORE	PD_SYSTEM	PD_AO	PD_MEM	PD_RF	PD_WUP
	flash (1024 kB)	-	-	RAM_X1 (4 kB)	-	-
	PUF RAM (2 kB)	-	-	RAM_X2 (4 kB)	-	-
	-	-	-	RAM_X3 (4 kB)	-	-
	-	-	-	RAM_00 (16 kB)	-	-
	-	-	-	RAM_10 (32 kB)	-	-
memories (cont'd)	-	-	-	RAM_20 (64 kB)	-	-
	-	-	-	RAM_BLEDATA (16 kB)	-	-
	-	-	-	RAM_BLESEQL(0.5 kB)	-	-

13.1.3 Power modes

Power modes are controlled exclusively via a power API [Section 14.3](#) and the operating mode of the CPU.

There are six 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 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 remains 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 and registers maintain their internal states. All SRAM instances that are not configured to enter the retention state, stay in active mode and consume more power. Through the power profiles API, selected peripherals such as SPI, I²C, 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 at the cost of a longer wake-up time compared to deep-sleep mode. The power-down mode affects the entire system, the clock to the CPU and peripherals is shut down. 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. All registers lose their internal states except the ones located in the power domains PD_SYSTEM and PD_AO. Any SRAM instance that is not configured to maintain its internal state loses its content. On wake-up from power-down mode with CPU retention disabled, the part reboots.
When CPU retention option is selected, the internal state of the CPU0, ROM patch unit, AHB security controller, and PRINCE are maintained. When a wake-up event occurs after power-down mode with CPU retention, code execution resumes from where it left off. It is the responsibility of the customer application to reconfigure all modules in the power domain core PD_CORE (whose states have not been retained (that

is, SDMA and all Flexcomm products except for Flexcomm0). Through the power profiles API, selected peripherals such as Flexcomm0 Interface, SPI, I²C, USART, GINT0, RTC, and OS event timer, can be enabled to wake up the system. On wake-up, the part reboots.

- Deep power-down

Deep-power down mode shuts down virtually all on-chip power consumption. However, it requires a much 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 PMC, the RTC, and the OS event timer. On wake-up, the part reboots.

- Power-off

Power-off mode shuts down to the entire chip except for the RESET_N pin and the wake-up pin (PIO0_21). No clock sources are active on-chip.

The MCX W23 can wake up from power-off mode via the RESET_N pin or the special wake-up pin (PIO0_21). SRAM is power-gated and loosing internal states. On wake-up, the part reboots.

[Table 136](#) summarizes the power states of the different power domains according to the power modes.

Table 136. Power modes

	PD_CORE	PD_SYSTEM	PD_AO	PD_MEM	PD_RF	PD_WUP
ACTIVE	ON	ON	ON	ON	ON	ON
SLEEP	ON	ON	ON	ON	ON	ON
DEEP-SLEEP	ON	ON	ON	ON/OFF	ON/OFF	ON
POWER-DOWN	OFF	ON	ON	ON/OFF	ON/OFF	ON
DEEP POWER- DOWN	OFF	OFF	ON	OFF	OFF	ON
POWER-OFF	OFF	OFF	OFF	OFF	OFF	ON

13.1.4 Peripheral configuration in reduced power modes

Table 137. Peripheral reduced power modes

Peripherals	Reduced power modes					
Name	Description	Sleep	Deep sleep	Power down	Deep power down	Power off
DCDC	DCDC converter	software configured (ON/BYPASS/OFF)	software configured (ON/BYPASS/OFF)	software configured (ON/BYPASS/OFF)	software configured (ON/BYPASS/OFF)	OFF
RTC	real-time clock	software configured	software configured	software configured	software configured	OFF
BOD 1	VBAT brownout detector	software configured	software configured	software configured	software configured	OFF
BOD 2	VBAT brownout detector	software configured	software configured	software configured	software configured	OFF
FRO1M	1 MHz free running oscillator	software configured	software configured	software configured	OFF	OFF

Table 137. Peripheral reduced power modes...continued

Peripherals	Reduced power modes					
Name	Description	Sleep	Deep sleep	Power down	Deep power down	Power off
FRO192M	192 MHz free running oscillator. Provides the main clock for the Arm and the SPIFI interface (divided down from the currently selected on-chip FRO192M oscillator).	software configured	software configured	OFF	OFF	OFF
FRO32K	32.768 kHz free running oscillator	software configured	software configured	software configured	software configured	OFF
XTAL32K	32.768 kHz crystal oscillator	software configured	software configured	software configured	software configured	OFF
XTAL32M	32 MHz crystal oscillator	software configured	software configured	software configured	OFF	OFF
LDO_MEM	regulator for PD_MEM power domain	ON	retention/OFF	retention/OFF	OFF	OFF
LDO_DIG	regulator for PD_CORE and PD_SYSTEM power domains	ON	ON/retention (mode depends on high frequency deep sleep mode - HFDSM)	retention	OFF	OFF
BLE_SLP_TMR	Bluetooth Low Energy sleep timer	software configured	software configured	software configured	OFF	OFF
BLE_XO_32M	Bluetooth Low Energy 32 MHz crystal oscillator	software configured	software configured	software configured	OFF	OFF
TRNG	true random number generator	software configured	software configured	OFF	OFF	OFF
ROM	ROM	ON	OFF	OFF	OFF	OFF

13.1.5 Wake-up process

The part always wakes up to the active mode. To wake up from the reduced power modes, the user must configure the wake-up source. Each reduced power mode supports its own wake-up sources and must be configured accordingly (see [Table 141](#)).

13.2 Functional description

13.2.1 Power management

The MCX W23 supports various 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 five special

modes of processor power reduction with different peripherals running, sleep mode, deep-sleep mode, power-down, deep-power-down mode, and power-off mode. The power-mode configure API activates them (see [Section 14.3](#)).

Note: The debug mode is not supported in sleep mode, deep-sleep mode, power-down mode, or deep-power down mode.

13.2.2 Active mode

In active mode, the AHB/CPU clock clocks the CPU, memories, and peripherals.

The chip is in active mode after reset. The reset values of the PDRUNCFG0 and AHBCLKCTRL registers determine the default power configuration is. The power configuration can be changed during runtime.

13.2.2.1 Power configuration in active mode

The following configuration choices determine the power consumption in active mode:

- The AHBCLKCTRL registers control which memories and peripherals are enabled (see the sections "AHB clock control 0 (AHBCLKCTRL0)" and "AHB clock control 1 (AHBCLKCTRL1)" in the Register description ([Ref. 5](#)). To save power, functions that the application does not require must be turned off. If specific times are known when certain functions are not required, they can be turned OFF temporarily and turned back ON again as necessary.
- The power to various analog blocks (oscillators and the BOD circuits) can be controlled individually through the power control API functions `POWER_PeripheralPowerOn()` and `POWER_PeripheralPowerOff()`. As with clock controls, these blocks must be turned OFF if the application does not require them. If turned OFF, it takes time before these blocks can be used again after being turned ON.
- The clock source for the system clock can be selected from the 12 MHz FRO (default), the 32 MHz FRO, the 1 MHz FRO, or the 32.768 kHz RTC clock (see [Figure 2](#) and related registers). The system clock frequency can be selected (see [Section 4.2.1](#) and other clocking-related sections). 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) saves power.
- Several peripherals use individual peripheral clocks with their own clock dividers. If the base clock is still required for another function, the peripheral clocks can be shut down through the corresponding clock divider registers.
- The power library provides an easy way to optimize power consumption depending on CPU load and performance requirements (see [Section 14](#)).

13.2.3 Sleep mode

In sleep mode, the system clock to the CPU is stopped and execution of instructions is suspended until a reset or an interrupt occurs.

Peripheral functions, if selected to be clocked in the AHBCLKCTRL registers, continue operation during sleep mode. To cause the processor to resume execution, they may generate interrupts. Sleep mode eliminates dynamic power that the processor itself, the memory systems and related controllers, and the internal buses use. The processor state and registers, peripheral registers, and internal SRAM values are maintained. 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 [Section 14](#)).

13.2.3.1 Power configuration in sleep mode

The same settings as in active mode determine the power consumption in sleep 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.

13.2.3.2 Programming sleep mode

The following steps must be performed to enter sleep mode:

1. In the NVIC, enable all interrupts that are required to wake up the relevant CPU.
2. To reflect any functions that must be ON or OFF during sleep mode, use power API functions `POWER_PeripheralPowerOn()` and `POWER_PeripheralPowerOff()`.
3. Call the power API `POWER_EnterSleep()` (see [Section 14](#)).

13.2.3.3 Wake-up from sleep mode

When an interrupt enabled by the NVIC arrives at the processor or a reset occurs, a CPU sleep mode is exited automatically. 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 microcontroller enters the default configuration in active mode.

13.2.4 Deep-sleep mode

In deep-sleep mode, the system clock to the CPUs is disabled as in sleep mode. Analog blocks are powered down by default. However, they can be selected to keep running through the power API. 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 that the analog peripherals use and all dynamic power that the CPU, its memory systems and related controllers, and internal buses use. The CPUs, state and registers, peripheral registers, and internal SRAM values are maintained. The logic levels of the pins remain static.

GPIO pin interrupts, GPIO group interrupts, and selected peripherals such as SPI, I²C, USART, WWDT, RTC, standard counter/timers, DMA, and BOD can be left running in deep-sleep mode. Power to the FRO1 MHz and the FRO 192 MHz, RTC oscillator, and the low-frequency oscillators (FRO 32.768 kHz and XO 32.768 kHz) is software configurable.

13.2.4.1 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, only contribute to the power consumption to the extent that external sources clock them. 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 [Section 14](#)).

13.2.4.2 Programming deep-sleep mode

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

1. On power-up, the BOD1 and BOD2 are enabled to assist the DCDC during start-up. To configure the BOD1 and BOD2 reset and interrupt generation in deep-sleep mode, power API functions are available.
2. To enable the analog peripherals and wake-up sources/events, call the power API `POWER_EnterDeepSleep` with its parameters (see [Section 14](#)).

13.2.4.3 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 [Section 19](#)).
- When enabled during the reduced power mode via the power API, using an interrupt from a block such as the watchdog interrupt, BOD1/BOD2 interrupt or RTC interrupt.
- Using a reset from the `RESET_N` pin or `WWDT`.
- Using a wake-up signal from any of the serial peripherals that operate in deep-sleep mode.
- GPIO group interrupt signal.
- RTC alarm signal or wake-up signal (see [Section 27](#)).
- OS event timer (see [Section 31](#)).

All wake-up interrupts (and sources) must be enabled via the power API.

13.2.5 Power-down mode

Power-down mode eliminates power used by almost all analog and digital peripherals. The FRO 192 MHz is disabled. The flash memory is also disabled. When CPU retention is enabled, the processor state and some critical peripherals, like the security controller, are retained. Internal SRAM values can be maintained. The logic levels of the pins remain static.

GPIO group interrupts, selected serial peripherals in Flexcomm0 (SPI, I²C, USART), RTC, BODs, OS event timers, and Bluetooth Low Energy sleep timer can be left running.

13.2.5.1 Power configuration in power-down mode

The number of SRAM instances which remain enabled (retention mode) primarily determine the power consumption in power-down mode is. Serial peripherals in Flexcomm0 and pin interrupts configured to wake up, only contribute to the dynamic power consumption to the extent that external sources clock them. All wake-up events (other than reset) must be enabled via the power API. In addition, any analog block (the BODs, XO 32.768 kHz, and FRO 32.768 kHz) must be explicitly enabled through a power API function (see [Section 14](#)).

13.2.5.2 Programming power-down mode

To enter power-down mode, the following steps must be performed:

1. On power-up, the BOD1 and BOD2 are enabled to assist the DCDC during start-up. To configure the BOD1 and BOD2 reset and interrupt generation in power-down mode, power API functions are available.
2. To enable the analog peripherals (BODs, DCDC, 32.768 kHz Xtal, or 32.768 kHz FRO) and select the wake-up sources, call the power API `POWER_EnterPowerdown()` with CPU retention mode parameter and the peripheral parameter (see [Section 14](#)).

13.2.5.3 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_N pin.
- Using a wake-up signal from any of the serial peripherals in Flexcomm0.
- GPIO group interrupt signal.
- RTC alarm signal or wake-up signal (see [Section 27](#)).
- BOD1 or BOD2 interrupt or reset signal.
BOD1 or BOD2 reset signal.
- OS event timer (see [Section 31](#)).
- Bluetooth Low Energy sleep timer

All wake-up interrupts (and sources) must be enabled via the power API.

13.2.6 Deep power-down mode

In deep-power down mode, power and clocks are shut off to the entire chip except the PMC, the RTC, the BODs, and the OS event timer.

During deep-power down mode, the contents of the SRAM and registers (other than the ones in the PMC, the RTC, the BODs, and OS event timer) are not retained. All functional pins are tristated in deep-power down mode, except the wake-up pin and the RESET_N pin.

13.2.6.1 Power configuration in deep power-down mode

Deep power-down mode has the following configuration options (via the low power API)

- Wake-up pins
- 32.768 kHz clock source for RTC and OS event timer
- BOD1 and BOD2 enable
- DCDC bypass

All clocks, the core, and all peripherals are powered down. Only the PMC, the RTC, and the OS event timer are powered with the associated clock source, 32.768 kHz FRO or 32.768 kHz crystal.

13.2.6.2 Programming deep-power down mode using the RTC for wake-up

To enter deep-power down mode when using the RTC for waking up, the following steps must be performed:

1. Set up the RTC high-resolution timer. Write to the RTC VAL register. If enabled, it starts the high-resolution timer. Another option is to use the 1 Hz alarm timer.
2. Call the power API function (see [Section 14](#)).

13.2.6.3 Programming deep-power down mode using the OS event timer for wake-up

To enter deep-power down mode when using the OS event timer for waking up, the following steps must be performed:

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 [Section 14](#)).

13.2.6.4 Programming deep-power down mode using the wake-up pin for wake-up

To enter deep-power down mode when using the wake-up pin for waking up, the following steps must be performed:

1. Call the power API function (see [Section 14](#)).

13.2.6.5 Wake-up from deep power-down mode

Wake-up from deep-power down can be accomplished via:

- The RESET_N pin
- RTC
- OS event timer
- The wake-up pin (PIO0_21)
- BOD1 or BOD2 interrupt or reset signal
BOD1 or BOD2 reset signal

When a deep-power down wake-up event occurs, the part goes through the entire reset process:

- The PMC turns on the on-chip voltage regulator. When the core voltage reaches the power-on-reset (POR) trip point, a system reset is triggered and the chip boots.
- All registers are in their reset states.

13.2.7 Power-off mode

In power-off mode, power and clocks are shut off to the entire chip with the exception of the Wake-up logic.

During power-off mode, the contents of the SRAM and registers are not retained. All functional pins are tri-stated in power-off mode, except the wake-up pin (PIO0_21) and the RESET_N pin.

14 Power profiles/power control API

The power profiles and power control APIs can be implemented using the power library from the SDK software package available on NXP.com.

14.1 Features

- Simple APIs to control power consumption and wake-up in all low-power modes: sleep, deep-sleep, power-down, deep power-down, and power-off mode.
- Prepares the part to enter low-power modes: sleep, deep-sleep, power-down, deep power-down, and power-off mode.
- Manages power consumption for sleep and active modes.

14.2 General description

Control of device power consumption or entry to low-power modes can be configured through simple calls to the low power API.

APIs exist to:

- Set up reduced power modes.

Terminology: RAM instances can have following states in the different power modes:

- Active state
RAM keeps data state and can be accessed.
- Retention state
RAM keeps data state.
- Shutdown state
RAM loses data state.

14.3 Power-related API descriptions

Power APIs provide functions to configure the system during active and sleep power modes.

Table 138. Power API calls

Function prototype	API description	Section
void POWER_Init(void);	Function to initialize the library.	None
void POWER_Reset(void);	Function to reset device. This will result in a full boot cycle.	None
uint32_t POWER_GetLibVersion(void);	Function to return the library version.	None
void POWER_PeripheralPowerOff(power_config_bit_t power_config_bit);	Function to disable the power of an analog peripheral when the device is in active mode.	Section 14.3.1
void POWER_PeripheralPowerOn(power_config_bit_t power_config_bit);	Function to enable the power of an analog peripheral when the device is in active mode.	Section 14.3.1
void POWER_DisableSRAM(uint32_t sram_ctrl);	Function to power off certain SRAM segments.	Section 14.3.2
void POWER_EnableSRAM(uint32_t sram_ctrl);	Function to power on certain SRAM segments. By default all SRAM segments are on.	Section 14.3.2
uint32_t POWER_GetSRAMConfig(void);	Function to return power status of all SRAM segments.	Section 14.3.2

Table 138. Power API calls...continued

Function prototype	API description	Section
void POWER_SetBod1Level(bod_level_t level);	Function to configure the detection level of BOD1.	Section 14.3.3
void POWER_SetBod2Level(bod_level_t level);	Function to configure the detection level of BOD2.	Section 14.3.3
void POWER_ConfigureBOD1(bod_action_t action);	Function to configure the BOD1 action.	Section 14.3.3
void POWER_ConfigureBOD2(bod_action_t action);	Function to configure the BOD2 action.	Section 14.3.3
void POWER_ClearBOD1Interrupt(void);	Function to clear the BOD1 interrupt.	Section 14.3.3
void POWER_ClearBOD2Interrupt(void);	Function to clear the BOD2 interrupt.	Section 14.3.3
bool POWER_IsBOD1Active(void);	Function to retrieve the BOD1 status.	Section 14.3.3
bool POWER_IsBOD2Active(void);	Function to retrieve the BOD2 status.	Section 14.3.3
dc_dc_mode_t POWER_DCDC_GetSupply Mode(void);	Function that returns the Supply Mode.	Section 14.3.4
status_t POWER_DCDC_SetSupply Mode(dc_dc_mode_t mode);	Function that sets the Supply mode.	Section 14.3.4
void POWER_DCDC_Disable(void);	Function to disable DC/DC convertor.	Section 14.3.4
void POWER_DCDC_Enable(void);	Function to enable DC/DC convertor.	Section 14.3.4
void POWER_DCDC_ConfigureBuck Output(dc_dc_buck_output_level_t output_lvl);	Function to configure output level of DC/DC convertor when configured in buck mode.	Section 14.3.4
uint16_t POWER_DCDC_GetStartup Duration(void);	Function to retrieve the time in microseconds it took for the DC/DC to have stable output level.	Section 14.3.4
reset_cause_t POWER_GetResetCause(void);	Function to retrieve the reset cause.	None
void POWER_XTAL32K_ConfigureCaps(uint8_t capIn, uint8_t capOut);	Function to configure the caps of the 32KHz xtal.	Section 14.3.5
void POWER_XTAL32K_ConfigureReference OutputCurrent(uint8_t gm);	Function to configure the reference output current of the 32KHz xtal.	Section 14.3.5
void POWER_XTAL32K_ConfigureOscillator Amplitude(uint8_t ampl);	Function to configure the oscillator amplitude of the 32KHz xtal.	Section 14.3.5
void POWER_XTAL32K_EnableBypass(bool enable);	Function to configure 32k xtal to bypass mode.	Section 14.3.5
void POWER_XTAL32K_SetStartup Time(uint32_t startupTime);	Function to configure XTAL oscillator start up time.	Section 14.3.5
void POWER_XTAL32M_BLE_GateEnable(bool enable);	Function to gate 32M BLE xtal.	Section 14.3.6
status_t POWER_XTAL32M_SetStartup Time(uint8_t startupTime);	Function to configure 32MHz XTAL oscillator start up time.	Section 14.3.6

Low power APIs provide functions to configure the system into the different low-power modes: sleep, deep sleep, power down, and deep power down.

Table 139. Low power API calls

Function prototype	API description	Section
void POWER_EnterSleep(void);	This API makes the CPU enter sleep mode.	Section 14.3.7
status_t POWER_EnterDeepSleep (uint32_t exclude_from_pd, uint64_t wakeup_interrupts);	This API configures the chip then enters deep-sleep mode.	Section 14.3.8
status_t POWER_EnterPowerDown(uint32_t exclude_from_pd, uint64_t wakeup_interrupts, uint32_t cpu_retention_ctrl);	This API configures the chip then enters power-down mode.	Section 14.3.9
status_t POWER_EnterDeepPowerDown(uint32_t exclude_from_pd, uint64_t wakeup_interrupts, uint32_t wakeup_io_ctrl);	This API configures the chip then enters deep power-down mode.	Section 14.3.10
status_t POWER_EnterPowerOff(uint32_t exclude_from_pd, uint32_t wakeup_io_ctrl);	This API configures the chip then enters power-off mode.	Section 14.3.11

The `exclude_from_pd` parameter defines which analog peripherals are not powered down and can wake up the chip from a low-power mode. The excluded peripherals remain running in low-power mode. For example, if the WWDT must remain active in low-power mode, the FRO-1 MHz oscillator must be running.

The `exclude_from_pd` parameter is a 32-bit value that corresponds with the definition in [Table 140](#). The values from the enumeration `low_power_config_bit_t` can be ORed together to create a parameter value. For each bit field:

- '0': The module is powered down during deep-sleep mode.
- '1': The module is running during deep-sleep mode.

[Table 140](#) lists which peripherals can be excluded from powering down for each power mode.

Table 140. Parameter `exclude_from_pd`

Peripherals	Description	low_power_config_bit_t	Deep-sleep mode	Power-down mode	Deep power-down mode	Power-off mode
DCDC	DCDC converter	KLOWPOWERCFG_DCDC	YES	YES	YES	YES
DCDC_Bypass	DCDC bypass switch. Only activated if DCDC is in power down.	KLOWPOWERCFG_DCDC_BYPASS	YES	YES	YES	YES
BOD1	brownout detector 1	KLOWPOWERCFG_BOD1	YES	YES	YES	NO
BOD2	brownout detector 2	KLOWPOWERCFG_BOD2	YES	YES	YES	NO
FRO1M	1 MHz free running oscillator	KLOWPOWERCFG_FRO1M	YES	YES	NO	NO
FRO192M	192 MHz free running oscillator	KLOWPOWERCFG_FRO192M	YES	NO	NO	NO
FRO32K	32.768 kHz free running oscillator	KLOWPOWERCFG_FRO32K	YES	YES	YES	NO
XTAL32K	32.768 kHz crystal oscillator	KLOWPOWERCFG_XTAL32K	YES	YES	YES	NO
BLE_WUP	Bluetooth Low Energy wake-up timer. When running, the PMC activates the XO32M during the wake-up sequence.	KLOWPOWERCFG_BLE_WUP	YES	YES	NO	NO
HFDSM	high-frequency deep-sleep mode. The wake-up peripheral can run on a clock up to 32 MHz. Enabling this	KLOWPOWERCFG_HFDSM	YES	NO	NO	NO

Table 140. Parameter exclude_from_pd...continued

Peripheral	Description	low_power_config_bit_t	Deep-sleep mode	Power-down mode	Deep power-down mode	Power-off mode
	configuration comes at expense of higher current consumption.					
TRNG	true random number generator. Only activated when HFDSM is enabled.	kLOWPOWERCFG_TRNG	YES	NO	NO	NO

The wakeup_interrupts parameter defines which peripheral interrupts can be a wake-up source during low-power modes.

[Table 141](#) describes, for each low-power mode, if an interrupt can be the source for a wake-up.

The wake-up peripherals have a maximal clock speed, depending on the power mode:

- Deep-sleep mode:
Maximum peripheral speed is 1 MHz.
If a peripheral is required to run on 12 or 32 MHz the HFDSM must be called in the exclude_from_pd parameter. Other exceptions are identified in [Table 141](#).
- Power-down mode
Maximum peripheral speed is 1 MHz. Exceptions are identified in [Table 141](#).
- Deep-power down mode
32 kHz
- Power-off
No clock is running.

The customer application must configure all used wake-up peripherals with a correct frequency corresponding to the selected power mode.

[Table 141](#) lists which interrupt sources can be used to wake up for each power mode.

Table 141. Parameter wakeup_interrupts

Wake-up source	Description	wakeup_irq_t	Deep-sleep mode	Power-down mode	Deep power-down mode	Power-off mode
WAKEUP_SYS	watchdog timer, BODs, and flash interrupt	k WAKEUP_SYS	YES	NO	NO	NO
WAKEUP_SDMA0	system DMA	k WAKEUP_DMA0	YES	NO	NO	NO
WAKEUP_GPIO_GLOBALINT0	GINT0	k WAKEUP_GINT0	YES	YES	NO	NO
WAKEUP_GPIO_INT0_0	GPIO	k WAKEUP_PIN_INT0	YES	NO	NO	NO
WAKEUP_GPIO_INT0_1	GPIO	k WAKEUP_PIN_INT1	YES	NO	NO	NO

Table 141. Parameter wakeup_interrupts...continued

Wake-up source	Description	wakeup_irq_t	Deep-sleep mode	Power-down mode	Deep power-down mode	Power-off mode
WAKEUP_GPIO_INT0_2	GPIO	k WAKEUP_PIN_INT2	YES	NO	NO	NO
WAKEUP_GPIO_INT0_3	GPIO	k WAKEUP_PIN_INT3	YES	NO	NO	NO
WAKEUP_UTICK	microtick timer	k WAKEUP_UTICK	YES (maximum 32.768 kHz)	NO	NO	NO
WAKEUP_MRT	multirate timer	k WAKEUP_MRT	YES	NO	NO	NO
WAKEUP_CTIMER0	standard counter/timer 0	k WAKEUP_CTIMER0	YES (maximum 32.768 kHz)	NO	NO	NO
WAKEUP_CTIMER1	standard counter/timer 1	k WAKEUP_CTIMER1	YES (maximum 32.768 kHz)	NO	NO	NO
WAKEUP_SCT	SCTimer/PWM	k WAKEUP_SCT	YES	NO	NO	NO
WAKEUP_CTIMER3	standard counter/timer 3	k WAKEUP_CTIMER3	YES (maximum 32.768 kHz)	NO	NO	NO
WAKEUP_FLEXCOMM0	USART, SPI, I ² C	k WAKEUP_FLEXCOMM0	YES	YES	NO	NO
WAKEUP_FLEXCOMM1	USART, SPI, I ² C	k WAKEUP_FLEXCOMM1	YES	NO	NO	NO
WAKEUP_FLEXCOMM2	USART, SPI, I ² C	k WAKEUP_FLEXCOMM2	YES	NO	NO	NO
WAKEUP_BLE_LL	Bluetooth Low Energy link layer	k WAKEUP_BLE_LL	NO	NO	NO	NO
WAKEUP_BLE_SLP_TMR	Bluetooth Low Energy sleep timer	k WAKEUP_BLE_SLP_TMR	YES	YES	NO	NO

Table 141. Parameter wakeup_interrupts...continued

Wake-up source	Description	wakeup_irq_t	Deep-sleep mode	Power-down mode	Deep power-down mode	Power-off mode
WAKEUP_Code WDG	code watchdog timer	k WAKEUP_WDT	YES	NO	NO	NO
WAKEUP_BOD1	BOD1	k WAKEUP_BOD1	YES	NO	NO	NO
WAKEUP_BOD2	BOD2	k WAKEUP_BOD2	YES	NO	NO	NO
WAKEUP_RTC	RTC	k WAKEUP_RTC	YES	YES	YES	NO
WAKEUP_WAKE_DSLP	wake-up from deep-sleep mode	k WAKEUP_WAKE_DSLP	YES	NO	NO	NO
WAKEUP_GPIO_INT0_4	GPIO	k WAKEUP_PIN_INT4	YES	NO	NO	NO
WAKEUP_GPIO_INT0_5	GPIO	k WAKEUP_PIN_INT5	YES	NO	NO	NO
WAKEUP_GPIO_INT0_6	GPIO	k WAKEUP_PIN_INT6	YES	NO	NO	NO
WAKEUP_GPIO_INT0_7	GPIO	k WAKEUP_PIN_INT7	YES	NO	NO	NO
WAKEUP_CTIMER2	standard counter/timer 2	k WAKEUP_CTIMER2	YES (maximum 32.768 k Hz)	NO	NO	NO
WAKEUP_CTIMER4	standard counter/timer 4	k WAKEUP_CTIMER4	YES (maximum 32.768 k Hz)	NO	NO	NO
WAKEUP_OS_EVENT_TIMER	OS event timer	k WAKEUP_OS_EVENT	YES	YES	YES	NO
WAKEUP_SPIFI	quad SPI flash interface	k WAKEUP_SPIFI	YES	NO	NO	NO
WAKEUP_SEC_GPIO_INT0_0	secure GPIO	k WAKEUP_SEC_	YES	NO	NO	NO

Table 141. Parameter wakeup_interrupts...continued

Wake-up source	Description	wakeup_irq_t	Deep-sleep mode	Power-down mode	Deep power-down mode	Power-off mode
		GPIO_INT0_IRQ0				
WAKEUP_SEC_GPIO_INT0_1	secure GPIO	k WAKEUP_SEC_GPIO_INT0_IRQ1	YES	NO	NO	NO
WAKEUP_PLU	programmable logic	k WAKEUP_PLU	YES	NO	NO	NO
WAKEUP_SEC_VIO	security violation	k WAKEUP_SEC_VIO	NO	NO	NO	NO
WAKEUP_TRNG	true random number generator	k WAKEUP_TRNG	YES	NO	NO	NO
WAKEUP_DMA1	DMA	k WAKEUP_DMA1	YES	NO	NO	NO
WAKEUP_WAKE_PAD	wake-up from WAKEUP pin interrupt PAD	k WAKEUP_WAKE_PAD	NO	NO	YES	YES

The wakeup_interrupts parameter is a 64-bit value. The values from the enumeration wakeup_irq_t can be ORed together to create a parameter value. For each bit field:

- '0': The associated peripheral is not a wake-up source during deep-sleep mode.
- '1': The associated peripheral is a wake-up source during deep-sleep mode.

Notes

- CPUs and system clock frequency are switched to FRO12MHz. Upon exit, the POWER_EnterDeepSleep, POWER_EnterPowerDown, and POWER_EnterDeepPowerDownpower APIs restore them.
- The POWER_EnterDeepSleep, POWER_EnterPowerDown, and POWER_EnterDeepPowerDown power APIs modify the CPU interrupt enable registers (NVIC → ISER). If in deep-sleep mode and in power-down mode with CPU retention or if the low-power mode is not entered (for example, due to a pending interrupt), they are restored.
- If any interrupt that is used as wake up source is pending, the system will not go to the low power mode. To avoid this, all interrupts that are configured as wakeup_interrupts will be briefly enabled in the NVIC so they can be handled just before entering the low power mode. This means that the application must have an interrupt handler installed for every interrupt that is used as a wakeup source. Special care must be given to WDT, BOD1 and BOD2 because they can be handled by the kWAKEUP_SYS interrupt or by their individual kWAKEUP_WDT, kWAKEUP_BOD1 or kWAKEUP_BOD2 interrupt. Please make sure that if using e.g. kWAKEUP_WDT that a handler is installed for WDT_IRQn, otherwise the default handler will be called when this interrupt is briefly enabled. This default handler will block the system forever.

- The non-maskable interrupt (NMI) is disabled and, before calling this function, its configuration is restored if in deep-sleep mode and in power-down mode with CPU retention.
- The HARD FAULT handler must execute from SRAM and not from flash (The hard fault handler must initiate a full chip reset).

14.3.1 POWER_PeripheralPowerOff/On

These routines disable or enable the power of an analog peripheral when the device is in active mode.

Table 142. POWER_PeripheralPowerOff and POWER_PeripheralPowerOn API routine

Routine	POWER_PeripheralPowerOff and POWER_PeripheralPowerOn
SDK prototype	void POWER_PeripheralPowerOff(power_config_bit_t power_config_bit); void POWER_PeripheralPowerOn(power_config_bit_t power_config_bit);
input parameter	power_config_bit: peripheral for which to disable or enable the power
result	none
description	Function to disable or enable the power of an analog peripheral when the device is in active mode.

The value for the power_config_bit must be one of the enumeration power_config_bit_t:

- kPOWERCFG_DCDC
- kPOWERCFG_BIAS
- kPOWERCFG_BOD1
- kPOWERCFG_BOD2
- kPOWERCFG_FRO192M
- kPOWERCFG_FRO1M
- kPOWERCFG_FRO32K
- kPOWERCFG_XTAL32K
- kPOWERCFG_BLE
- kPOWERCFG_RNG

14.3.2 POWER_Dis/EnableSRAM

These routines disable or enable the power to certain SRAM segments.

Table 143. POWER_DisableSRAM and POWER_EnableSRAM API routine

Routine	POWER_DisableSRAM and POWER_EnableSRAM
SDK prototype	void POWER_DisableSRAM(uint32_t sram_ctrl); void POWER_EnableSRAM(uint32_t sram_ctrl);
input parameter	sram_ctrl: mask of sram_ctrl_bit_t bits to power off or on.
result	none
description	Function to power off or on certain SRAM segments. By default all SRAM segments are on.

The values for the sram_ctrl must be ORed together from the enumeration sram_ctrl_bit_t:

- kSRAM_CTRL_RAMX0
- kSRAM_CTRL_RAMX1
- kSRAM_CTRL_RAMX2
- kSRAM_CTRL_RAMX3
- kSRAM_CTRL_RAM00
- kSRAM_CTRL_RAM10
- kSRAM_CTRL_RAM20
- kSRAM_CTRL_RAMBLEDATA
- kSRAM_CTRL_RAMBLESEQ

14.3.3 POWER_Bod

These routines control the behavior of the Brown Out Detectors.

Table 144. POWER_SetBodLevel API routines

Routine	POWER_SetBod1Level and POWER_SetBod2Level
SDK prototype	void POWER_SetBod1Level(bod_level_t level); void POWER_SetBod2Level(bod_level_t level);
input parameter	level: BOD detect level
result	none
description	Function to configure the detection level of the BOD. The BOD will be active while the voltage is below the detection level. The BOD is configured to kBOD_NO_ACTION by this function before changing the detection level. It must be followed by POWER_ConfigureBOD if reset or interrupt is required.

The value for the level must be one of the enumeration bod_level_t:

- kBOD_LEVEL_1100mv
- kBOD_LEVEL_1125mv
- kBOD_LEVEL_1150mv
- ... continues in steps of 25mV ...
- kBOD_LEVEL_3050mv
- kBOD_LEVEL_3075mv
- kBOD_LEVEL_3100mv

Table 145. POWER_ConfigureBOD API routines

Routine	POWER_ConfigureBOD1 and POWER_ConfigureBOD2
SDK prototype	void POWER_ConfigureBOD1(bod_action_t action); void POWER_ConfigureBOD2(bod_action_t action);
input parameter	action: the action to take when a BOD is detected.
result	none
description	Function to configure the BOD action. Note that BOD is disabled automatically when BOD reset is triggered but the settings remain. Set the desired threshold voltage using POWER_SetBodLevel before enabling BOD interrupt or BOD reset.

The values for the action must be one of the enumeration bod_action_t:

- kBOD_NO_ACTION
- kBOD_RESET
- kBOD_INTERRUPT

Configuring the BOD to trigger an interrupt does not enable the interrupt in the NVIC. Make sure to install an appropriate interrupt handler. The BOD interrupt can be handled by the specific BOD1_IRQHandler (enabled by BOD1_IRQn) or BOD2_IRQHandler (enabled by BOD2_IRQn). It can also be handled by the combined WDT_BOD_IRQHandler (enabled by WDT_BOD_IRQn).

The BOD will stay active as long as the voltage is below the threshold level. This means that the interrupt will keep triggering if it is not disabled in the interrupt handler.

Table 146. POWER_ClearBODInterrupt API routine

Routine	POWER_ClearBOD1Interrupt and POWER_ClearBOD2 Interrupt
SDK prototype	void POWER_ClearBOD1Interrupt(void); void POWER_ClearBOD2Interrupt(void);
input parameter	none
result	none
description	Function to clear the BOD interrupt output to the NVIC. It does not clear the interrupt in the NVIC. The BOD will stay active as long as the voltage is below the threshold level so if the BOD interrupt is not disabled, it will trigger again immediately in this case.

Table 147. POWER_IsBODActive API routine

Routine	POWER_IsBOD1Active and POWER_IsBOD2Active
SDK prototype	bool POWER_IsBOD1Active(void); bool POWER_IsBOD2Active(void);
input parameter	none
result	true as long as the BOD detects a voltage below the threshold.
description	Function to retrieve the BOD status.

14.3.4 POWER_DcDc

This routines can be used to configure or get information about the DC/DC converter.

Table 148. POWER_DCDC_GetSupplyMode API routine

Routine	POWER_DCDC_GetSupplyMode
SDK prototype	dc_dc_mode_t POWER_DCDC_GetSupplyMode(void);
input parameter	none
result	Supply Mode.
description	Function that returns the Supply Mode.

Table 149. POWER_DCDC_SetSupplyMode API routine

Routine	POWER_DCDC_SetSupplyMode
SDK prototype	status_t POWER_DCDC_SetSupplyMode(dcdc_mode_t mode);
input parameter	Supply Mode.
result	status_t: - kStatus_Success: Successfully set the supply mode. - kStatus_Fail: The supply mode is not valid and is not set.
description	Function that sets the Supply mode

The values for the supply mode must be one of the enumeration dcdc_mode_t:

- kDCDC_MODE_HV_SM: HV supplied
- kDCDC_MODE_XR_SM_SS: HV and LV supplied and equal (Single Supply)
- kDCDC_MODE_XR_SM_DS: HV and LV supplied and not equal (Dual Supply)

Table 150. POWER_DCDC_Disable and POWER_DCDC_Enable API routine

Routine	POWER_DCDC_Disable and POWER_DCDC_Enable
SDK prototype	void POWER_DCDC_Disable(void); void POWER_DCDC_Enable(void);
input parameter	none
result	none
description	Functions to disable or enable the DC/DC convertor. When the device is using an external supply mode, the DC/DC convertor should be disabled. When supply mode is kDCDC_MODE_HV_SM, the DCDC can't be disabled.

Table 151. POWER_DCDC_ConfigureBuckOutput API routine

Routine	POWER_DCDC_ConfigureBuckOutput
SDK prototype	void POWER_DCDC_ConfigureBuckOutput(dcdc_buck_output_level_t output_lvl);
input parameter	output_lvl: The output level of the DC/DC convertor.
result	none
description	Function to configure output level of DC/DC convertor when configured in buck mode. Can only be used supply mode kDCDC_MODE_HV_SM is used.

The values for the output level must be one of the enumeration dcdc_buck_output_level_t:

- kDCDC_BUCK_OUTPUT_1100mv
- kDCDC_BUCK_OUTPUT_1200mv
- kDCDC_BUCK_OUTPUT_1300mv
- kDCDC_BUCK_OUTPUT_1400mv
- kDCDC_BUCK_OUTPUT_1500mv
- kDCDC_BUCK_OUTPUT_1600mv
- kDCDC_BUCK_OUTPUT_1700mv
- kDCDC_BUCK_OUTPUT_1800mv

Table 152. POWER_DCDC_GetStartupDuration API routine

Routine	POWER_DCDC_GetStartupDuration
SDK prototype	uint16_t POWER_DCDC_GetStartupDuration(void);
input parameter	none
result	Value that indicates how long it took for DC/DC convertor to have a stable output in microseconds.
description	Function to retrieve the time it took for the DC/DC to have stable output level. Note that the measurement circuit saturates at 0xFFFF (65535 us). The times it takes for the DC/DC to stabilize the output can be used to fine-tune the BLE wake up timer. The BLE stack needs to enable 'correction' of the BLE wakeup timer based on how long it takes for the DC/DC to stabilize.

14.3.5 POWER_XTAL32K

These routines are all related to the configuration of the 32KHz crystal.

Table 153. POWER_XTAL32K_ConfigureCaps API routine

Routine	POWER_XTAL32K_ConfigureCaps
SDK prototype	void POWER_XTAL32K_ConfigureCaps(uint8_t capIn, uint8_t capOut);
input parameter	capIn: The capIn setting capOut: The capOut setting
result	none
description	Function to configure the caps of the 32.768 KHz crystal.

Table 154. POWER_XTAL32K_ConfigureReferenceOutputCurrent API routine

Routine	POWER_XTAL32K_ConfigureReferenceOutputCurrent
SDK prototype	void POWER_XTAL32K_ConfigureReferenceOutputCurrent(uint8_t gm);
input parameter	gm: The reference output current selection setting (0, 1, 2 or 3)
result	none
description	Function to configure the reference output current of the 32.768 KHz crystal.

Table 155. POWER_XTAL32K_ConfigureOscillatorAmplitude API routine

Routine	POWER_XTAL32K_ConfigureOscillatorAmplitude
SDK prototype	void POWER_XTAL32K_ConfigureOscillatorAmplitude(uint8_t ampl);
input parameter	ampl: The oscillator amplitude setting (0, 1, 2 or 3)
result	none

Table 155. POWER_XTAL32K_ConfigureOscillatorAmplitude API routine...continued

Routine	POWER_XTAL32K_ConfigureOscillatorAmplitude
description	Function to configure the oscillator amplitude of the 32.768 KHz crystal

Table 156. POWER_XTAL32K_EnableBypass API routine

Routine	POWER_XTAL32K_EnableBypass
SDK prototype	void POWER_XTAL32K_EnableBypass(bool enable);
input parameter	enable: True to put xtal in bypass mode, false otherwise
result	none
description	Function to configure 32.768 KHz crystal oscillator to bypass mode. Caps will be set to 0 and XO32K block will be powered when bypass is enabled.

Table 157. POWER_XTAL32K_SetStartupTime API routine

Routine	POWER_XTAL32K_SetStartupTime
SDK prototype	void POWER_XTAL32K_SetStartupTime(uint32_t startup Time);
input parameter	delay: delay time in milliseconds
result	none
description	Function to configure 32.768 KHz crystal oscillator start up time. This function overrides the default start up time of 500 ms.

14.3.6 POWER_XTAL32M

These routines are all related to the configuration of the 32MHz crystal.

Table 158. POWER_XTAL32M_BLE_GateEnable API routine

Routine	POWER_XTAL32M_BLE_GateEnable
SDK prototype	void POWER_XTAL32M_BLE_GateEnable(bool enable);
input parameter	enable: True to pass clock to BLE block, false otherwise
result	none
description	Function to gate 32M BLE xtal

Table 159. POWER_XTAL32M_SetStartupTime API routine

Routine	POWER_XTAL32M_SetStartupTime
SDK prototype	status_t POWER_XTAL32M_SetStartupTime(uint8_t startup Time);
input parameter	startupTime: startup time in steps of 32 us.
result	status_t:

Table 159. POWER_XTAL32M_SetStartupTime API routine...continued

Routine	POWER_XTAL32M_SetStartupTime
	- kStatus_Success: The startup time was successfully configured. - kStatus_InvalidArgument: The startup time was not between 1 and 15.
description	Function to configure 32MHz XTAL oscillator start up time. This function overrides the default start up time of 288 us. The resolution is 32 us. Only values from 1 until 15 are allowed (1=32us, 15=480us).

14.3.7 POWER_EnterSleep

This routine puts the MCX W23 in sleep mode (when this routine is called, the activity of the CPU is stopped).

Table 160. POWER_EnterSleep API routine

Routine	POWER_EnterSleep
SDK prototype	void POWER_EnterSleep(void);
input parameter	none
result	none
description	Function to enter sleep: stop CPU, continues execution after wake up by any interrupt.

14.3.8 POWER_EnterDeepSleep

This routine prepares the MCX W23 to enter the 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.

Before calling POWER_EnterDeepSleep(), the application must configure the RAMs with the API POWER_EnableSRAM() and POWER_DisableSRAM().

- If the SRAM instance was enabled (active state) in active mode, it enters retention state during deep-sleep mode. After wake-up from deep-sleep, it returns back to active state.
- If the SRAM instance was disabled (shutdown state), it stays in the shutdown state during deep-sleep mode after wake-up from deep-sleep mode.

Table 161. POWER_EnterDeepSleep API routine

Routine	POWER_EnterDeepSleep
SDK prototype	status_t POWER_EnterDeepSleep (uint32_t exclude_from_pd, uint64_t wakeup_interrupts);
input parameter	Param0: exclude_from_pd: mask of power domain peripheral bits to specify which not to power down during the low power mode. See Table 140 to understand which values are valid for this power mode. Param1: wakeup_interrupts: mask of wake up interrupt bits to specify which interrupt sources can cause a wake up from the low power mode. See Table 141 to understand which values are valid for this power mode.
result	status_t: - kStatus_Success: Successfully woken up from low power mode. - kStatus_InvalidArgument: Invalid argument(s), low power mode was never entered. Reasons why this function could return kStatus_InvalidArgument: - Non existing low_power_config_bit_t value passed to exclude_from_pd

Table 161. POWER_EnterDeepSleep API routine...continued

Routine	POWER_EnterDeepSleep
	– FRO32K or XTAL32K not enabled (not added to exclude_from_pd), when BLE_WUP is excluded from pd (BLE can only remain on, if it has a clock) – FRO192M excluded from pd, but HFDSM not – BOD Resets enabled (using POWER_ConfigureBOD) but BOD not powered (using POWER_PeripheralPowerOn) – utick timer clock enabled, while going to deep sleep, but HFDSM not added to exclude from pd list. – FRO1M clock routed to ctimerX, while going to deep sleep, but HFDSM not added to exclude from pd list.
description	Function to enter deep sleep mode: stop CPU and power down flash, active srams put in retention, continues execution after wakeup. If any interrupt that is used as wake up source is pending, the system will not go to the low power mode. To avoid this, all interrupts that are configured as wakeup_interrupts will be briefly enabled in the NVIC so they can be handled just before entering the low power mode. This means that the application must have an interrupt handler installed for every interrupt that is used as a wakeup source. Special care must be given to WDT, BOD1 and BOD2 because they can be handled by the kWAKEUP_SYS interrupt or by their individual kWAKEUP_WDT, kWAKEUP_BOD1 or kWAKEUP_BOD2 interrupt. Please make sure that if using e.g. kWAKEUP_WDT that a handler is installed for WDT_IRQn, otherwise the default handler will be called when this interrupt is briefly enabled. This default handler will block the system forever.

14.3.8.1 Param0: exclude_from_pd

The exclude_from_pd parameter defines which analog peripherals are not powered down and can wake up the chip from deep-sleep. The excluded peripherals remain running in deep-sleep mode. For example, if the WWDT is to remain active in deep-sleep mode, the FRO-1 MHz oscillator must be running.

The exclude_from_pd parameter is a 32-bit value that corresponds to the definition of the table [Table 140](#).

14.3.8.2 Param1: wakeup_interrupts

The wakeup_interrupts parameter defines which peripheral interrupts can be a wake-up source during deep-sleep mode.

[Table 141](#) shows which interrupt can be a wake-up source during deep-sleep mode.

If any interrupt that is used as wake up source is pending, the system will not go to the low power mode. To avoid this, all interrupts that are configured as wakeup_interrupts will be briefly enabled in the NVIC so they can be handled just before entering the low power mode. This means that the application must have an interrupt handler installed for every interrupt that is used as a wakeup source. Special care must be given to WDT, BOD1 and BOD2 because they can be handled by the kWAKEUP_SYS interrupt or by their individual kWAKEUP_WDT, kWAKEUP_BOD1 or kWAKEUP_BOD2 interrupt. Please make sure that if using e.g. kWAKEUP_WDT that a handler is installed for WDT_IRQn, otherwise the default handler will be called when this interrupt is briefly enabled. This default handler will block the system forever.

14.3.9 POWER_EnterPowerDown

This routine prepares the MCX W23 to enter the 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.

Before calling POWER_EnterPowerDown(), the application must configure the RAMs with the API POWER_EnableSRAM() and POWER_DisableSRAM().

- If the SRAM instance was enabled (active state) in active mode, it enters the retention state during power-down mode. After wake-up from power-down, it returns to the active state.
- If the SRAM instance was disabled (shutdown state), it stays in the shutdown state during power-down and after wake-up from power-down.

Table 162. POWER_EnterPowerDown API routine

Routine	POWER_EnterPowerDown
SDK prototype	status_t POWER_EnterPowerDown (uint32_t exclude_from_pd, uint64_t wakeup_interrupts, uint32_t cpu_retention_ctrl);
input parameter	Param0: exclude_from_pd: mask of power domain peripheral bits to specify which not to power down during the low power mode. See Table 140 to understand which values are valid for this power mode. Param1: wakeup_interrupts: mask of wake up interrupt bits to specify which interrupt sources can cause a wake up from the low power mode. See Table 141 to understand which values are valid for this power mode. Param2: cpu_retention_ctrl: 0 = CPU retention is disabled / 1 = CPU retention is enabled, all other values are RESERVED
result	status_t: • kStatus_Success: Successfully woken up from low power mode if CPU retention was enabled. If CPU retention was not enabled, function will never return because application reboot is expected. • kStatus_InvalidArgument: Invalid argument(s), low power mode was never entered. Reasons why this function could return kStatus_InvalidArgument: – Non existing low_power_config_bit_t value passed to exclude_from_pd – FRO32K or XTAL32K not enabled (not added to exclude_from_pd), when BLE_WUP is excluded from pd (BLE can only remain on, if it has a clock) – BOD Resets enabled (using POWER_ConfigureBOD) but BOD not powered (using POWER_PeripheralPowerOn) – Invalid wake up interrupt used for this power mode.
description	Function to enter power down mode: power down flash and CPU, active srams put in retention, continues execution after wakeup if CPU retention is enabled and resets after wake up if CPU retention is disabled. If any interrupt that is used as wake up source is pending, the system will not go to the low power mode. To avoid this, all interrupts that are configured as wakeup_interrupts will be briefly enabled in the NVIC so they can be handled just before entering the low power mode. This means that the application must have an interrupt handler installed for every interrupt that is used as a wakeup source.

Note: The user must ensure that the SRAM instance containing the stack used to call this software function is preserved during low-power mode via the POWER_EnableSRAM() function.

14.3.9.1 Param0: exclude_from_pd

The exclude_from_pd parameter defines which analog peripherals are not powered down and can wake up the chip from power-down.

[Table 140](#) shows which peripheral can be excluded from powering down.

14.3.9.2 Param1: wakeup_interrupts

The wakeup_interrupts parameter defines which peripheral interrupts can be a wake-up source during power-down mode.

[Table 141](#) shows which interrupt can be a wake-up source during power-down mode.

If any interrupt that is used as wake up source is pending, the system will not go to the low power mode. To avoid this, all interrupts that are configured as wakeup_interrupts will be briefly enabled in the NVIC so they can be handled just before entering the low power mode. This means that the application must have an

interrupt handler installed for every interrupt that is used as a wakeup source. Special care must be given to WDT, BOD1 and BOD2 because they can be handled by the kWAKEUP_SYS interrupt or by their individual kWAKEUP_WDT, kWAKEUP_BOD1 or kWAKEUP_BOD2 interrupt. Please make sure that if using e.g. kWAKEUP_WDT that a handler is installed for WDT_IRQn, otherwise the default handler will be called when this interrupt is briefly enabled. This default handler will block the system forever.

14.3.9.3 Param2: cpu_retention_ctrl

In power-down mode, the CPU state can be retained (cpu_retention_ctrl must be set to 0x1).

CPU state retention is implemented by shifting the CPU registers values inside SRAM instance RAM_X2, meaning that RAM_X2 must be kept in retention. Along with CPU, the states of AHB security controller and PRINCE registers values are also shifted in RAM_X2. The address range [0x0400_2000 - 0x0400_27FF] inside RAM_X2 is used, which means that, before calling the low power AP, any data in this area is lost.

After a wake-up event occurs, CPU resumes code execution.

When CPU state is retained, all SRAM instances that contain CPU variables (stack and heap) must also be retained.

14.3.10 POWER_EnterDeepPowerDown

This routine prepares the MCX W23 to enter the 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 163. POWER_EnterDeepPowerDown API routine

Routine	POWER_EnterDeepPowerDown
SDK prototype	status_t POWER_EnterDeepPowerDown (uint32_t exclude_from_pd, uint64_t wakeup_interrupts, uint32_t wakeup_io_ctrl);
input parameter	Param0: exclude_from_pd: mask of power domain peripheral bits to specify which not to power down during the low power mode. See Table 140 to understand which values are valid for this power mode. Param1: wakeup_interrupts: mask of wake up interrupt bits to specify which interrupt sources can cause a wake up from the low power mode. See Table 141 to understand which values are valid for this power mode. Param2: wakeup_io_ctrl: mask of wake up pin control bits (see wakeup_io_bits_t) to configure the wakeup pin.
result	status_t: - kStatus_InvalidArgument: Invalid argument(s), low power mode was never entered (in case of successful execution, function will never return because application reboot is expected). Reasons why this function could return kStatus_InvalidArgument: - Non existing low_power_config_bit_t value passed to exclude_from_pd - BOD Resets enabled (using POWER_ConfigureBOD) but BOD not powered (using POWER_PeripheralPowerOn) - Invalid wake up interrupt used for this power mode.
description	Function to enter deep power down mode: power down almost whole chip, reset/wake up by pin, RTC or OS Timer. If any interrupt that is used as wake up source is pending, the system will not go to the low power mode. To avoid this, all interrupts that are configured as wakeup_interrupts will be briefly enabled in the NVIC so they can be handled just before entering the low power mode. This means that the application must have an interrupt handler installed for every interrupt that is used as a wakeup source.

14.3.10.1 Param0: exclude_from_pd

The `exclude_from_pd` parameter defines which analog peripherals are not powered down and can wake up the chip from deep power-down mode.

[Table 140](#) shows which peripheral can be excluded from powering down mode.

14.3.10.2 Param1: wakeup_interrupts

The `wakeup_interrupts` parameter defines which peripheral interrupts can be a wake-up source during deep power-down mode.

[Table 141](#) shows which interrupt can be a wake-up source during deep power-down mode.

If any interrupt that is used as wake up source is pending, the system will not go to the low power mode. To avoid this, all interrupts that are configured as `wakeup_interrupts` will be briefly enabled in the NVIC so they can be handled just before entering the low power mode. This means that the application must have an interrupt handler installed for every interrupt that is used as a wakeup source. Special care must be given to WDT, BOD1 and BOD2 because they can be handled by the `kWAKEUP_SYS` interrupt or by their individual `kWAKEUP_WDT`, `kWAKEUP_BOD1` or `kWAKEUP_BOD2` interrupt. Please make sure that if using e.g. `kWAKEUP_WDT` that a handler is installed for `WDT_IRQn`, otherwise the default handler will be called when this interrupt is briefly enabled. This default handler will block the system forever.

14.3.10.3 Param2: wakeup_io_ctrl

The `wake_up_io_ctrl` parameter allows the configuration of the pin that can wake up the part from deep power-down mode.

Table 164. Parameter `wakeup_io_ctrl`

Symbol	Description	Value
WAKEUP_IO	enabling/disabling of wake-up pin in deep power-down modes	0: disabled 1: enabled
WAKEUP_IO_PUP	enabling/disabling pull-up on wake-up pin in deep power-down modes	0: disabled 1: enabled
WAKEUP_IO_PDN	enabling/disabling pull-down on wake-up pin in deep power-down modes	0: disabled 1: enabled
WAKEUP_IO_LVL	controls wake-up pin pull-up/pull-down level	0: low level 1: high level

14.3.11 POWER_EnterPowerOff

This routine prepares the MCX W23 to enter the power-off low-power mode. The API function configures the WAKEUP pin.

Table 165. `POWER_EnterPowerOff` API routine

Routine	<code>POWER_EnterPowerOff</code>
SDK prototype	<code>status_t POWER_EnterPowerOff (uint32_t exclude_from_pd, uint32_t wakeup_io_ctrl);</code>
input parameter	Param0: <code>exclude_from_pd</code>: mask of power domain peripheral bits to specify which not to power down during the low power mode. See Table 140 to understand which values are valid for this power mode. Param1: <code>wakeup_io_ctrl</code>: mask of wake up pin control bits (see <code>wakeup_io_bits_t</code>) to configure the wakeup pin.

Table 165. POWER_EnterPowerOff API routine...continued

Routine	POWER_EnterPowerOff
result	status_t: • kStatus_InvalidArgument: Invalid argument(s), low power mode was never entered (in case of successful execution, function will never return because application reboot is expected). Reasons why this function could return kStatus_InvalidArgument: – Non existing low_power_config_bit_t value passed to exclude_from_pd
description	Function to enter power off mode: power down almost whole chip, reset/wake up by pin only.

14.3.11.1 Param0: exclude_from_pd

The exclude_from_pd parameter defines which analog peripherals are not powered down and can wake up the chip from power-off mode.

[Table 140](#) shows which peripheral can be excluded from powering down.

14.3.11.2 Param1: wakeup_io_ctrl

The wake_up_io_ctrl parameter allows the configuration of the pin that can wake up the part from power-off mode (see table [Table 141](#)).

14.4 Functional description

For implementation considerations, see the power example application.

15 I/O pin configuration (IOCON)

The IOCON block is included on all MCX W23 devices.

Registers for pins that are not available on a specific package are reserved.

Note: Some functions, such as SCTimer/PWM inputs, frequency measure, and JTAG functions are not selected through IOCON. The connections for these functions are described in [Section 18](#) or the chapter for the specific function.

15.1 Features

All pins are standard (MFIO) port pins except PIO0_13 and PIO0_14 pins, which are combo I²C/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 I²C-bus speeds. Configuration options are different for these pins, as described in this chapter. For electrical details for these and other pins, see the MCX W23 data sheet ([Ref. 7](#)).

15.2 Basic configuration

Enable the clock to the IOCON in the AHBCLKCTRL0 register (see the section "AHB clock control 0 (AHBCLKCTRL0)" in the Register description ([Ref. 5](#)).

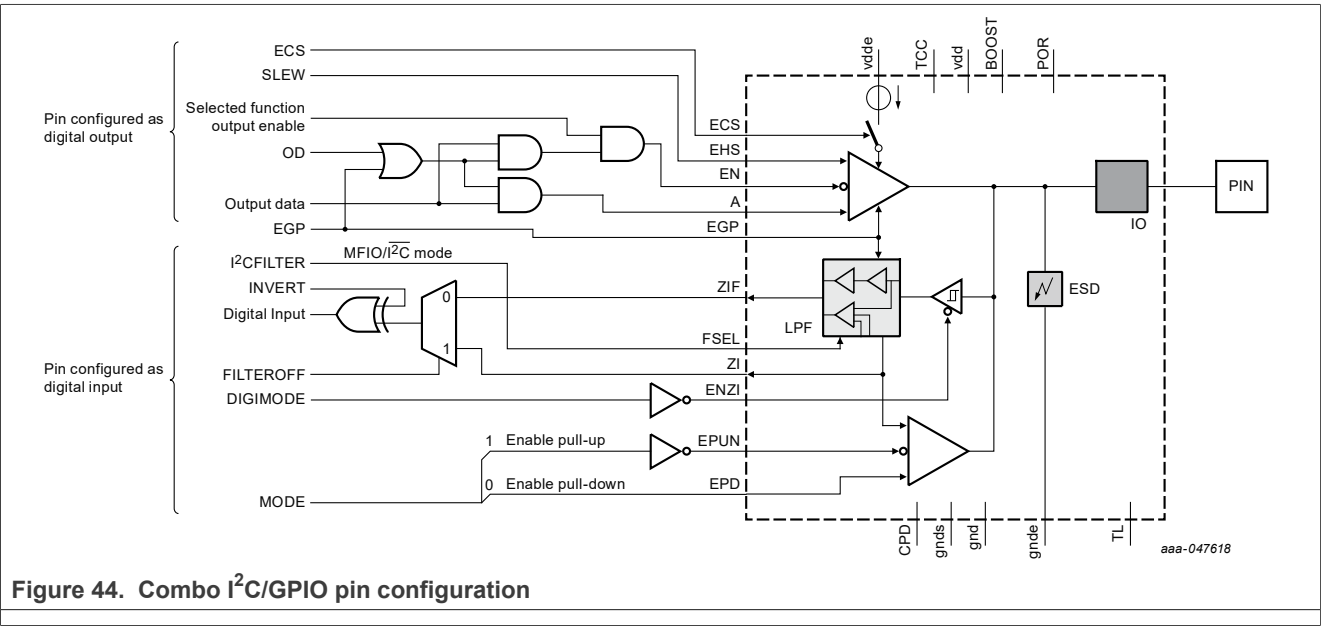
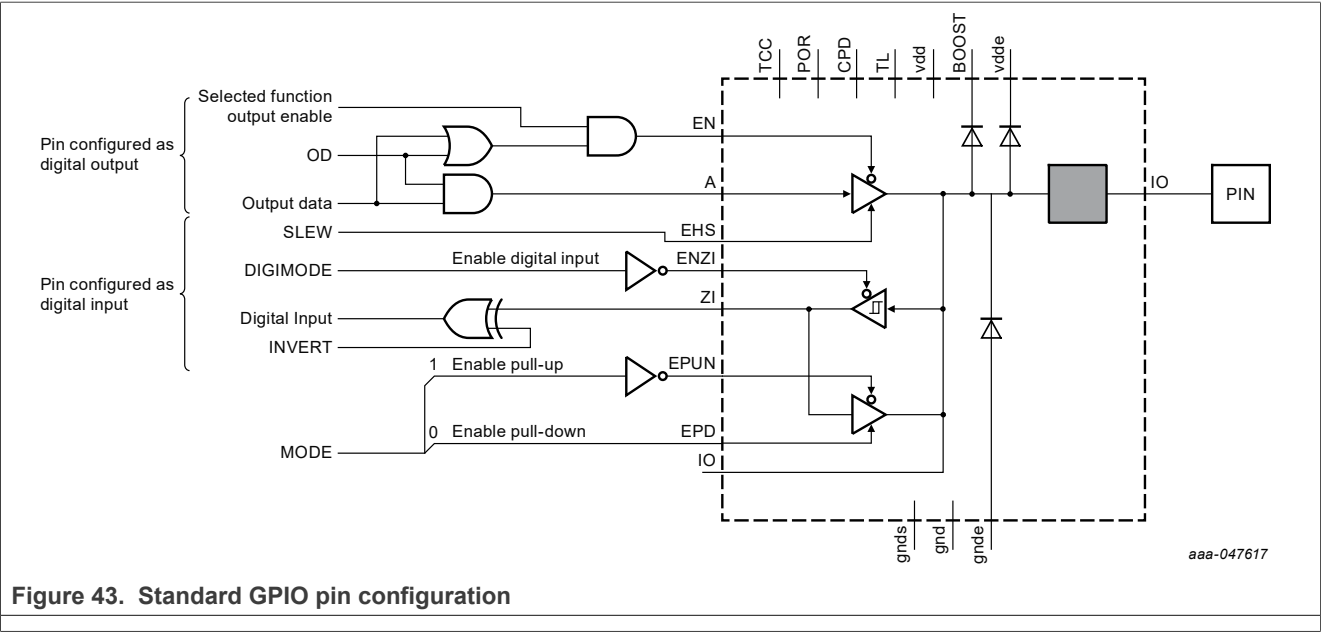
). When the pins are configured, the IOCON clock can be disabled to conserve power.

15.3 General description

15.3.1 Pin configuration

[Figure 43](#) shows the control of a standard GPIO pin.

[Figure 44](#) shows the control of a combo I²C/MFIO pin.



15.3.2 IOCON registers

The IOCON registers control the functions and properties of device pins. To select its function and characteristics, each GPIO pin has a dedicated control register. Each pin has a unique set of functional capabilities. Not all pin characteristics are selectable on all pins. For instance, pins that have an I²C function can be configured for different I²C-bus modes. Details of the IOCON registers are in [Section 15.3.2](#). The following sections describe specific characteristics of pins.

Multiple connections

It is possible to configure more than one pin to perform the same function, because a particular peripheral function may be allowed on more than one pin. If a peripheral output function is configured to appear on more

than one pin, it is routed to those pins. If a peripheral input function is defined as coming from more than one source, the values are logically combined, which can result in incorrect peripheral operation. So, care should be taken to avoid this situation.

15.3.2.1 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 PIO0_11 and PIO0_12 where default is FUNC = 1 (SWCLK and SWDIO special functions). For pins set to GPIO, the DIR registers in GPIO block determine if the pin is configured as an input or an output (see the section "Direction registers for all port GPIO pins (DIR0)" in the Register description ([Ref. 5](#))). For any special function, the pin direction is controlled automatically, depending on the function. The DIR registers have no effect for special functions.

15.3.2.2 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: PIO0_12/SWDIO, PIO0_20/ISP.
- Pull-up disable, Pull-down enable, Input enable: PIO0_11/SWCLK, PIO0_22/TMODE.

If the pin is high, the repeater mode enables the pull-up resistor. If the pin is low, it enables the pull-down resistor. If it is configured as an input and is not driven externally, it causes the pin to retain its last known state. Such a state retention is not applicable to the deep power-down mode. If a pin floats to an indeterminate state when temporarily not driven, the repeater mode may be used to prevent that a pin floats and uses significant power.

15.3.2.3 Hysteresis

The input buffer for digital functions has built-in hysteresis. For quantitative details, see the MCX W23 data sheet ([Ref. 7](#)).

15.3.2.4 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.

15.3.2.5 Analog/digital mode

When the MCX W23 is not in digital mode (DIGIMODE = 0), digital input from a pin is disabled and set to 0. In digital mode (DIGIMODE = 1), any analog pin functions are disabled. The MCX W23 does not have user-accessible analog pin functions.

15.3.2.6 Input filter

Some pins include a filter that can be selectively disabled by setting the FILTEROFF bit. It concerns the combo I²C/MFIO pins. The filter suppresses input pulses shorter than about 3 ns in MFIO mode and shorter than 10 ns or 50 ns in I²C mode, depending on the value of the I²CFILTER field.

15.3.2.7 Output slew rate

The SLEW bits of digital outputs that do not have to switch state must 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. It is important if analog accuracy is significant to the application. For more details, see the MCX W23 data sheet ([Ref. 7](#)).

15.3.2.8 I²C modes

Pins that support I²C with specialized pad electronics (PIO0_13 and PIO0_14) have additional configuration bits. To support the I²C variants, these pins have multiple configurations. They are not hardwired, so that the pins can be easily used for non-I²C functions. For recommended mode settings, see the section on the IO pin configuration in the Register description ([Ref. 5](#)).

For non-I²C operation, these pins can be configured as open-drain or push-pull, in the same way as for standard (MFIO) pins.

15.3.2.9 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 the DIR register of that port, 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 I²C/MFIO pins when I²C mode is selected but has the same effect as on a standard pin when in MFIO mode.

Note: The properties of a pin in this simulated open-drain mode are different from the pins of a true open-drain output.

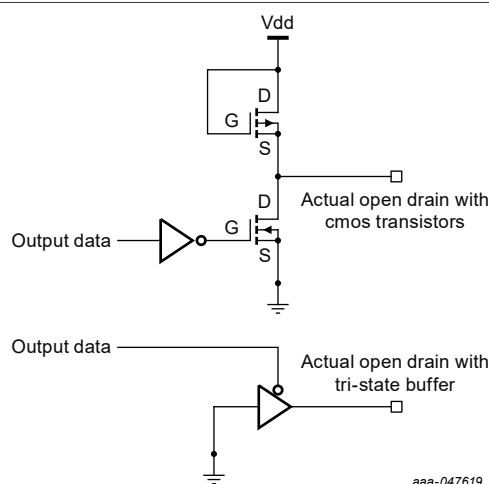


Figure 45. Open drain mode

16 General-purpose I/O (GPIO)

The GPIO registers support up to 23 pins. Depending on the device and package type, a subset of those pins may be available. The unused bits in GPIO registers are reserved (see [Table 166](#)).

Table 166. GPIO pins available

Package	Total number of GPIOs	GPIO port 0
MCXW236BIHN	23	PIO0_0 to PIO0_22
MCXW236BIUK	23	PIO0_0 to PIO0_22
MCXW236AIHN	23	PIO0_0 to PIO0_22
MCXW236AIUK	23	PIO0_0 to PIO0_22
MCXW235BIHN	23	PIO0_0 to PIO0_22
MCXW235BIUK	23	PIO0_0 to PIO0_22

16.1 Features

- Software can configure GPIO pins as input or output.
- Most GPIO pins default to tristate. However, a few pins are set to pull-up or pull-down as shown in [Section 15.3.2.2](#). 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 40.2.4](#)).

16.2 Basic configuration

For the GPIO port register, enable the clock to the GPIO port in the AHBCLKCTRL0 register (section "AHB clock control 0 (AHBCLKCTRL0)" in the Register description ([Ref. 5](#)).

16.3 General description

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

The GPIOs can be used as external interrupts together with the pin interrupt and group interrupt blocks (see [Section 19](#) and [Section 21](#)).

The GPIO port registers configure each GPIO pin as input or output. If the pin is configured as input, they read the state of each pin. If the pin is configured as output, they set the state of each pin.

16.4 Functional description

16.4.1 Reading pin state

Software can read the state of all GPIO pins except the pins selected for output in the I/O configuration logic. To read the state of a pin, it does not have to be selected for GPIO in I/O configuration. 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, halfword, or word from a word pin register.

- The state of multiple pins in a port can be read as a byte, halfword, or word from a port register.
- The state of a selected subset of the pins in a port can be read from a masked port (MPORT) register. Pins having a 1 in the mask register of the port 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 40.2.4](#)).

16.4.2 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 (the default except for 2 pins: PIO0_11 and PIO0_12).
2. The pin must be selected for output by a 1 in the DIR register of its port.

If either or both conditions 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 multibit value in programming languages).
- Writing to the PORT register of a port loads the output bits of all the pins written to.
- Writing to the MPORT register of a port loads the output bits of pins identified by zeros in corresponding positions of the mask register of a port.
- Writing ones to the SET register of a port sets output bits.
- Writing ones to the CLR register of a port CLR register clears output bits.
- Writing ones to a NOT register of a port 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 [Section 16.4.1](#) returns the state of pins, regardless of their direction or alternate functions.

16.4.3 Masked I/O

The MASK register of a port defines which of its pins must 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 the MASK register of a port 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. It 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 reenable them after the last operation that uses the MPORT or MASK register.

More efficiently, software can dedicate a semaphore to the MASK registers. It can 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.

16.4.4 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.

16.4.5 Recommended practices

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

- For initial setup after reset or reinitialization, 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. May require less write operations than SET and CLR.
- To read the state of one pin, read a byte pin or word pin register.
- To make a decision based on multiple pins, read and mask a PORT register.

17 Secure general-purpose I/O (SECGPIO)

The secure GPIO register supports up to the 23 pins on port 0. Each of the 23 secure GPIO pins is associated with a specific function (PIO0_SEC(i), i = 0 to 22) in the IOCON module.

17.1 Features

- Software can configure secure GPIO pins as input or output.
- 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.

17.2 Basic configuration

For the secure GPIO registers, enable the clock to secure GPIO in the SYSCON AHBCLKCTRL2 register (see the section "AHB clock control 2 (AHBCLKCTRL2)" in the Register description ([Ref. 5](#)).

17.3 General description

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

The secure GPIOs can be used as external interrupts together with the secure pin block (see [Section 20](#) and [Section 18](#)).

The secure GPIO registers configure each secure GPIO pin as input or output. If the pin is configured as input, it reads the state of each pin. If the pin is configured as output, it sets the state of each pin. When configured in output, for the bit value to be driven to the pin, FUNC = 7 must be set in the IOCON PIO0_n register (see [Section 16](#)).

17.4 Functional description

17.4.1 Reading pin state

Software can read the state of all secure GPIO pins. To read its state, a pin does not have to be selected for secure GPIO in I/O configuration. However, the input enable bit (DIGIMODE) of the pad must be set in I/O configuration. Otherwise, the pin state value is 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 mask register of the port reads as 0 from the MPORT register

17.4.2 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 the output bit of a pin to be driven onto the pin:

1. The pin must be selected for secure GPIO operation via IOCON (FUNC value must be 7).
2. The pin must be selected for output by a 1 in the DIR register.

If either or both conditions are not met, writing to the pin has no effect.

There are seven ways to change secure GPIO output bits:

- Writing to a byte pin register loads the output bit from the least significant bit.
- Writing to a word pin register loads the output bit with the OR of all of the bits written (this feature follows the definition of truth of a multi-bit value in programming languages).
- Writing to the PORT register loads the output bits of all the pins written to.
- Writing to the MPORT register loads the output bits of pins identified by zeros in corresponding positions of the MASK register.
- Writing ones to the SET register sets output bits.
- Writing ones to the CLR register clears output bits.
- Writing ones to the NOT register toggles/complements/inverts output bits.

The state of the output bits can be read from the SET register. Reading any of the registers described in [Section 17.4.1](#) returns the state of pins, regardless of their direction or alternate functions.

17.4.3 Masked I/O

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

Applications in which interrupts can result in masked secure GPIO operation or in task switching among tasks that do masked secure GPIO operation, must treat code that uses the mask register as a protected/restricted region. It 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 reenable 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.

17.4.4 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.

17.4.5 Recommended practices

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

- For initial setup after reset or reinitialization, 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. It can require less write operations than set and clr.
- To read the state of one pin, read a byte pin or word pin register.
- To make a decision based on multiple pins, read and mask the port register.

18 Input multiplexing (INPUTMUX)

Input multiplexing is present on all MCX W23 devices. Depending on the package, not all inputs from external pins may be available.

18.1 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; the FREQMECTRL register in [Section 11](#) controls this function

18.2 Basic configuration

When set up, no clocks are required for the input multiplexer to function. The system clock is only required to write to or read from the INPUT MUX registers. When the input multiplexer is configured, disable the clock to the INPUT MUX block in the AHBCLKCTRL register (see the section "AHB clock control 0" in the Register description ([Ref. 5](#))).

18.3 Pin description

The input multiplexer has no dedicated pins. However, all digital pins can be selected as inputs to the pin interrupts. Multiplexer inputs from external pins work independently of any other function assigned to the pin.

Table 167. INPUT MUX pin description

Pins	Peripheral	Section
Any existing pin on port 0	pin interrupts 0 to 7	Section 19
PIO0_3, PIO0_11, PIO0_12, PIO0_14	Frequency measure block	Section 11.4.1
SCT0_GPI [0:7] pin functions selected from IOCON register (See the pin descriptions in MCX W23 data sheet; Ref. 7).	SCTimer/PWM	Section 23

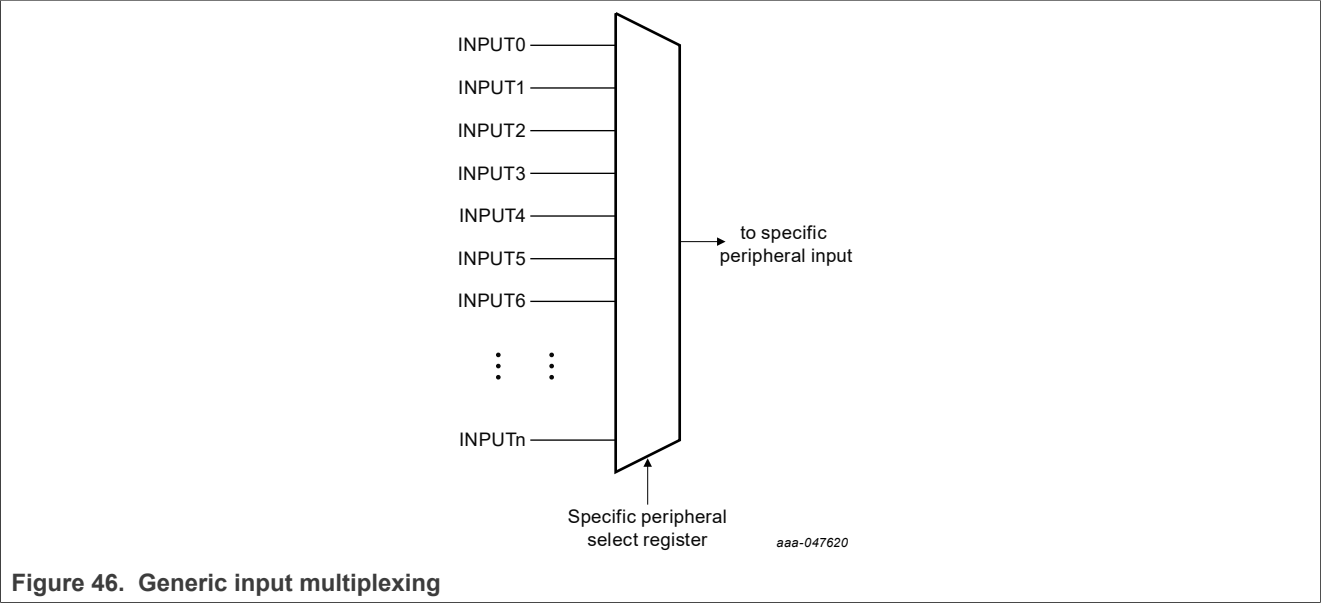
18.4 General description

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. The multiplexers 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 the capture inputs has an input multiplexer that can select a number of pin functions (if they are connected via IOCON) and some internal signals.

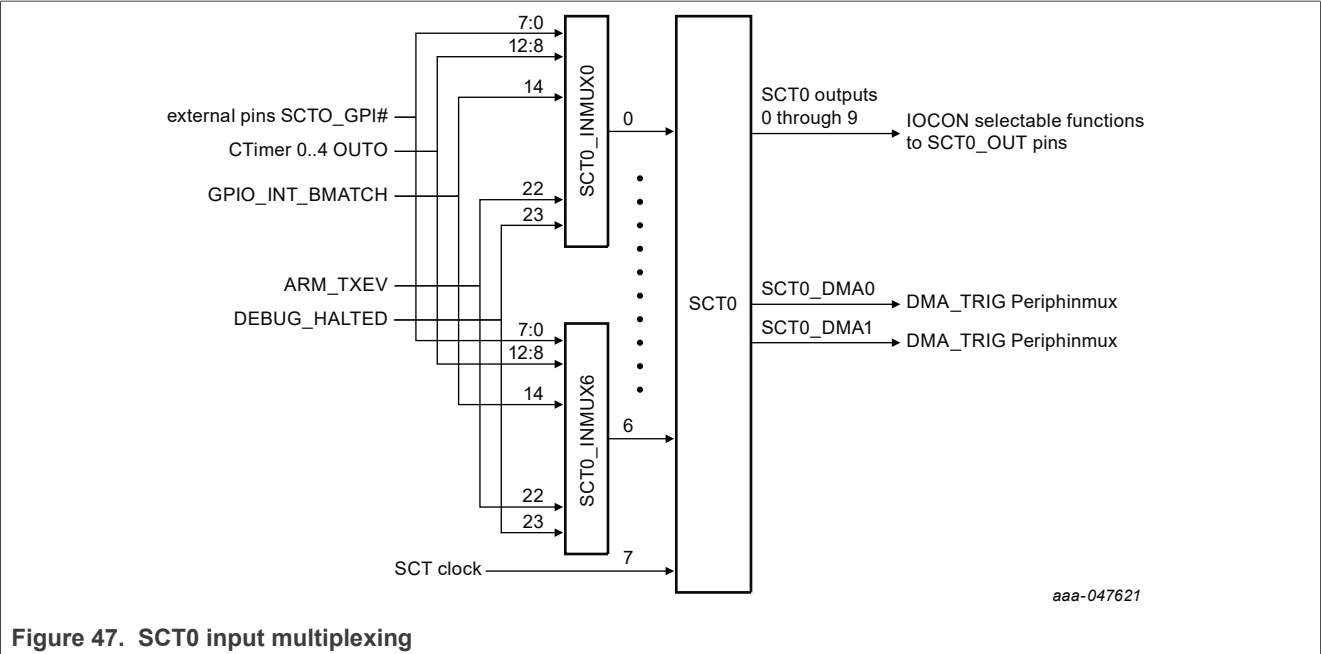
[Figure 46](#) shows a generic input multiplexer arrangement with n inputs. For example, regarding the CTimers, there are four of them for each timer, one for each capture input.

Some peripherals have a more complex arrangement and have detailed figures (see [Section 18.4.1](#) for SCT/PWM input multiplexing, [Section 18.4.2](#) for pin interrupt (PINT) multiplexing, and [Section 18.4.4](#) for DMA trigger multiplexing).



18.4.1 SCT0 input multiplexing

The input multiplexing for the SCT0 timer multiplexes between internal or external sources for each of its 7 outputs. These outputs and the SCT CLK are the 8 inputs of the SCT_TIMER. [Figure 47](#) shows the detail of this multiplexing.



18.4.2 Pin interrupt input multiplexing

The input multiplexing for the pin interrupts and pattern match engine multiplexes all existing pins from port 0.

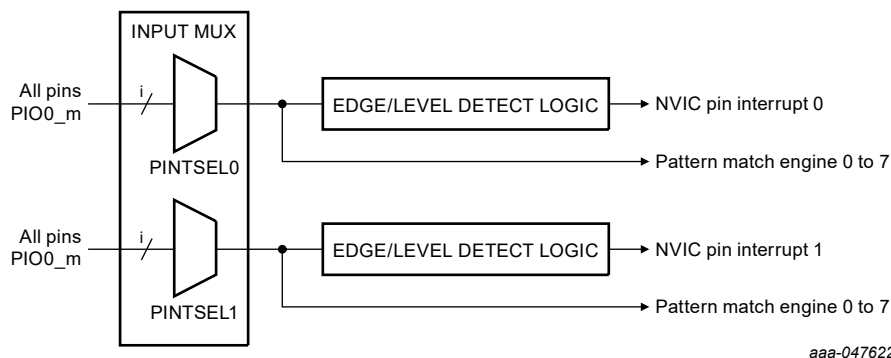


Figure 48. Pin interrupt multiplexing

18.4.3 Pin interrupt secure input multiplexing

The input multiplexing for the pin interrupts secure and pattern match engine multiplexes all existing pins from port 0 (with function 7, P0_SEC, selected).

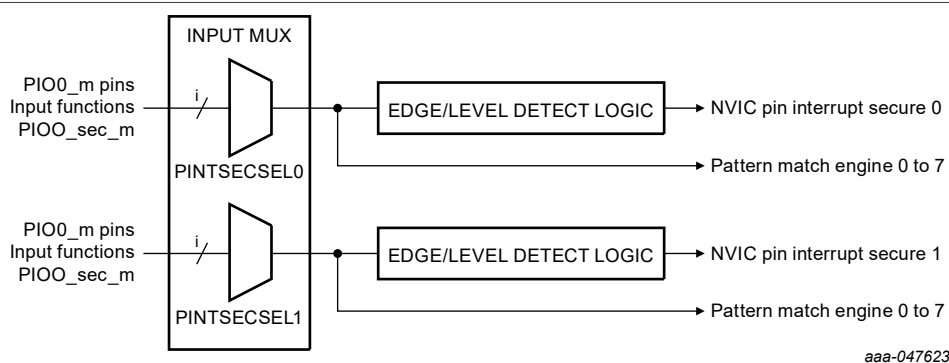
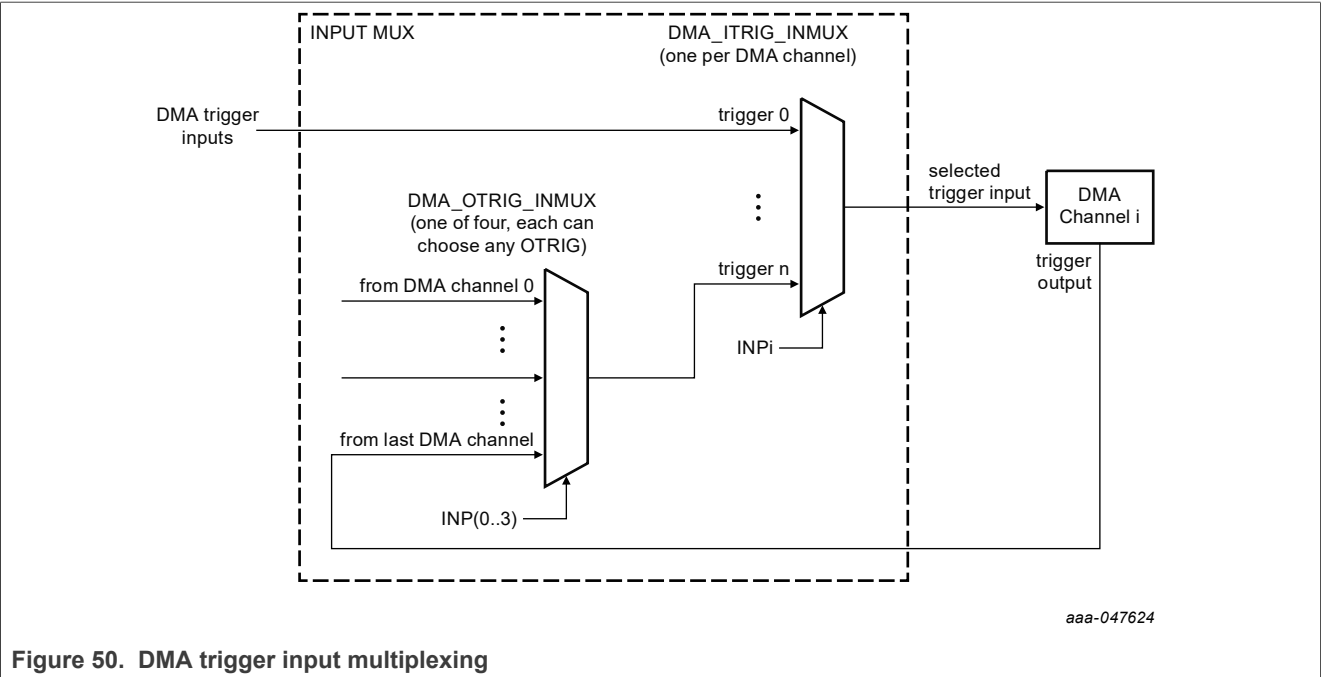


Figure 49. Pin interrupt secure multiplexing

18.4.4 DMA trigger input multiplexing

The trigger input multiplexing for each DMA controller is configured as shown in [Figure 50](#). In each DMA controller, four of these input triggers are selected from the DMA trigger outputs, controlled by the DMA_OTRIG_INMUX registers (section "DMA0 output trigger selection to become DMA0 trigger (DMA0_OTRIG_INMUX0 - DMA0_OTRIG_INMUX3)" in the Register description ([Ref. 5](#))).

The potential trigger selections for DMA0 are shown in the section "Trigger select register for DMA0 channel (DMA0_ITRIG_INMUXa)" and the potential trigger selections for DMA1 are shown in the section "Trigger select register for DMA1 channel (DMA1_ITRIG_INMUXa)" in the Register description ([Ref. 5](#))).



The two DMA controllers, DMA0 and DMA1, 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 which request.

19 Pin interrupt and pattern match (PINT)

The pin interrupt generator and the pattern match engine are available on all MCX W23 devices. PINT module uses standard GPIO functions as inputs.

19.1 Features

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

19.2 Basic configuration

- Pin interrupts
 - Select up to eight external interrupt pins from all digital port pins in the Input mux block (see the register descriptions in the section "Pin interrupt and pattern match (PINT)" of the Register description ([Ref. 5](#))). 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 the section "AHB clock control 0 (AHBCLKCTRL0)" in the Register description ([Ref. 5](#))).
 - In order to use the pin interrupts to wake up the part from deep-sleep mode, enable the pin interrupt wake-up feature using the low-power API.
- Pattern match engine
 - Select up to eight external pins from all digital port in the input mux block (see the register descriptions in the section "Pin interrupt and pattern match (PINT)" of the Register description ([Ref. 5](#))). 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 the section "AHB clock control 0 (AHBCLKCTRL0)" in the Register description ([Ref. 5](#))).
 - 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).

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

To configure pins as pin interrupts, follow these steps:

1. Determine the pins that serve as pin interrupts on the MCX W23 package. See the data sheet for determining the GPIO port pin number associated with the package pin.
2. For each pin interrupt, program the GPIO port pin number into one of the eight PINTSEL registers in the Input multiplexing block.
Note: 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.

When the pin interrupts or pattern match inputs are configured, the pin interrupt detection levels or the pattern match boolean expression can be set up. For the PINTSECSEL registers, see the section "Pin interrupt select register (PINTSELa)" in the Register description ([Ref. 5](#)).

Note: 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.

19.3 Pin description

The pin interrupt select registers in the input multiplexing determine the inputs to the pin interrupt and pattern match engine (see the section "Pin interrupt select register (PINTSELa)" in the Register description ([Ref. 5](#)).

19.4 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.

19.4.1 Pin interrupts

From all available GPIO pins, up to eight pins can be selected in the system control block to serve as external interrupt pins (see the register descriptions in the section "Pin interrupt and pattern match (PINT)" of the Register description ([Ref. 5](#)). 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.

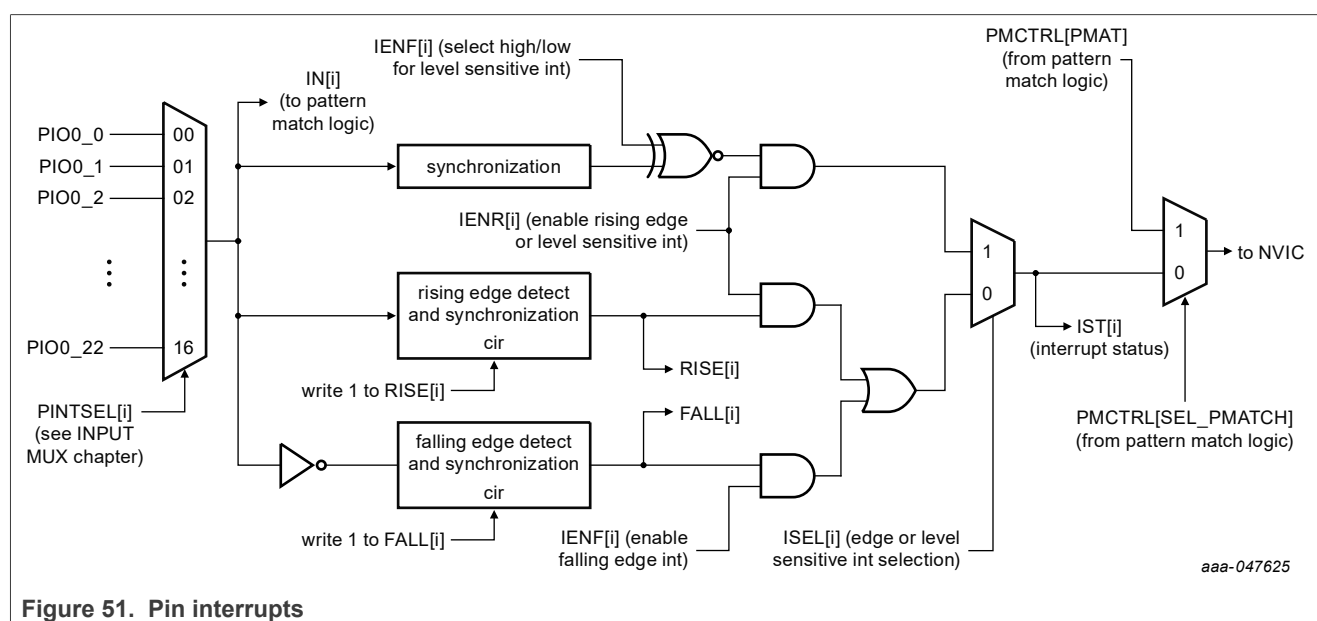


Figure 51. Pin interrupts

19.4.2 Pattern match engine

The pattern match feature allows complex boolean expressions to be constructed from the same set of eight GPIO pins that were selected for the GPIO pin interrupts. Each term in the boolean expression is implemented as one slice of the pattern match engine. A slice consists of an input selector and a detect logic that monitors

the selected input continuously and creates a HIGH output if the input qualifies as detected, that is, as true. Several terms can be combined to a minterm. When the minterm evaluates as true, a pin interrupt is asserted.

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

- Edge with memory (sticky):
A rising edge and/or a 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 53](#) shows the details of the edge detection logic for each slice.

To create a pin interrupt whenever a rising or falling edge occurs after a qualifying edge event, sticky events can be combined with non-sticky events.

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 (for details, see [Section 19.5.3](#)).

The connections between the pins and the pattern match engine are shown in [Figure 52](#). All pins that are inputs to the pattern match engine can be GPIO port pins or other pin function, depending on the IOCON configuration.

Note: *To operate, the pattern match feature requires clocks. It cannot not generate an interrupt or wake up the device during deep-sleep mode.*

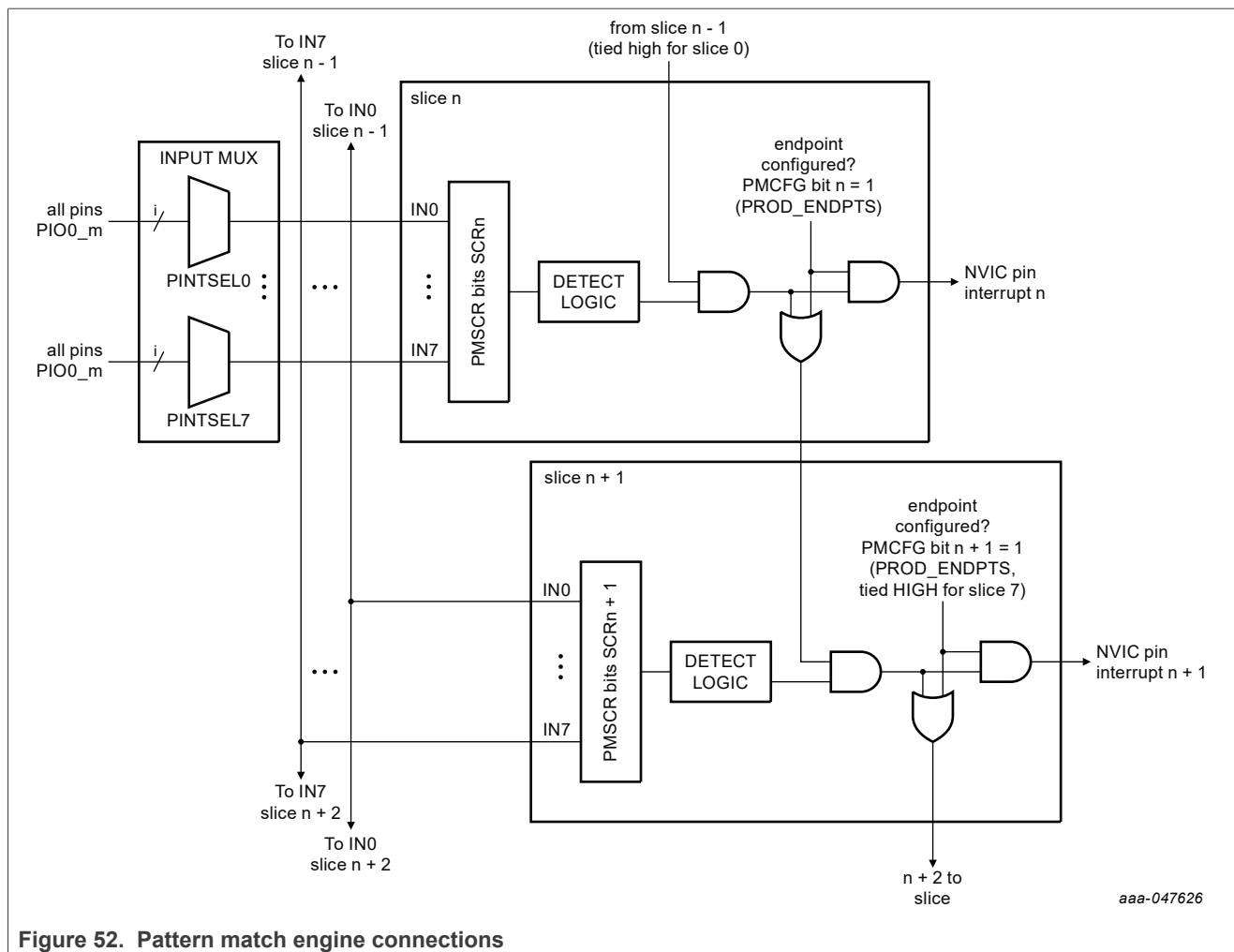


Figure 52. Pattern match engine connections

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

In addition, the pattern match module can be enabled to generate a receive event (RXEV) output to the Arm core (CPU0) when the entire boolean expression is true (that is, when any minterm is matched).

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

Note: Pattern matching cannot be used to wake up the part 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. When that minterm is matched, the interrupt request associated with the last bit slice for a particular minterm is asserted.

Figure 53 shows the bit slice.

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), which causes a transition out of the current state.

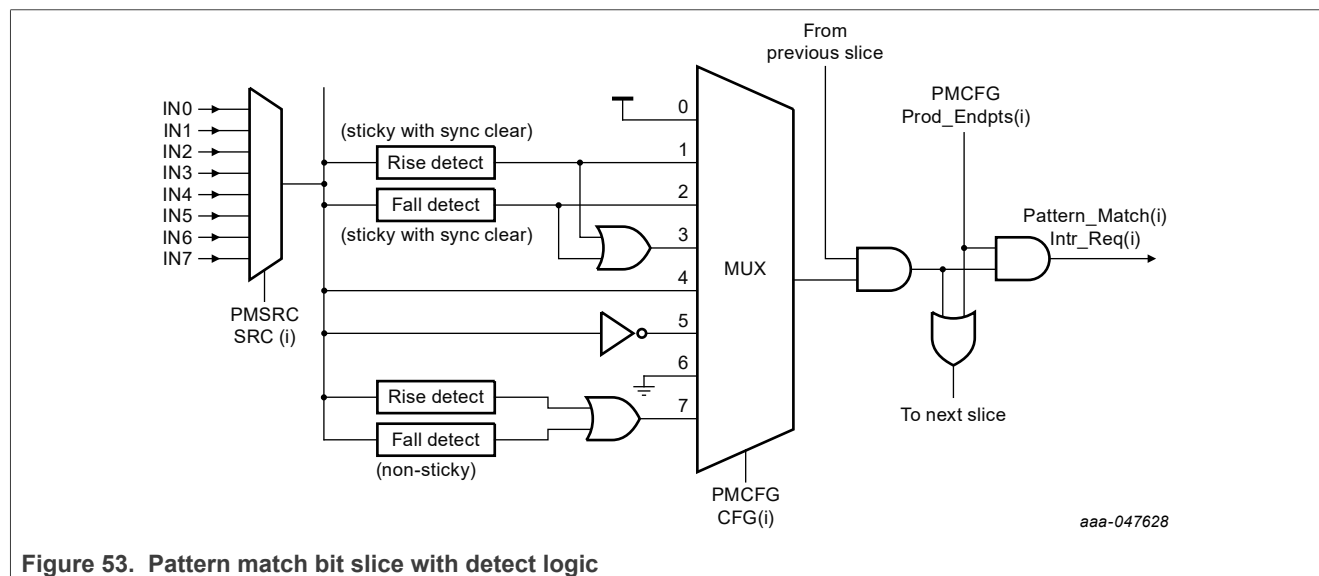


Figure 53. Pattern match bit slice with detect logic

19.4.2.1 Example

The following expression is specified through the registers PMSRC and PMCFG (see the sections "Pattern match interrupt bit-slice source register (PMSRC)" and "Pattern match interrupt bit slice configuration register (PMCFG)" in the Register description ([Ref. 5](#)).

```
(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, and so on, 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 is 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 is 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 is 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.

For more details, see [Section 19.5.2](#).

19.5 Functional description

19.5.1 Pin interrupts

In this interrupt facility, the pin interrupt select registers (PINTSEL0-7) identify up to 8 pins as interrupt sources. All registers in the pin interrupt block contain 8 bits, corresponding to the pins the PINTSEL0-7 registers call out. The ISEL register defines if each interrupt pin is edge-sensitive or level-sensitive. The RISE and FALL registers detect edges on each interrupt pin. They can be written to clear (and set) edge detection. The IST register indicates if each interrupt pin is requesting an interrupt. 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 168](#).

Table 168. 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

19.5.2 Pattern match engine example

Assumption: The desired boolean pattern to be matched is:

$(IN1) + (IN1 * IN2) + (\sim IN2 * \sim IN3 * IN6fe) + (IN5 * IN7ev)$, where:

IN6fe = (sticky) falling-edge on input 6

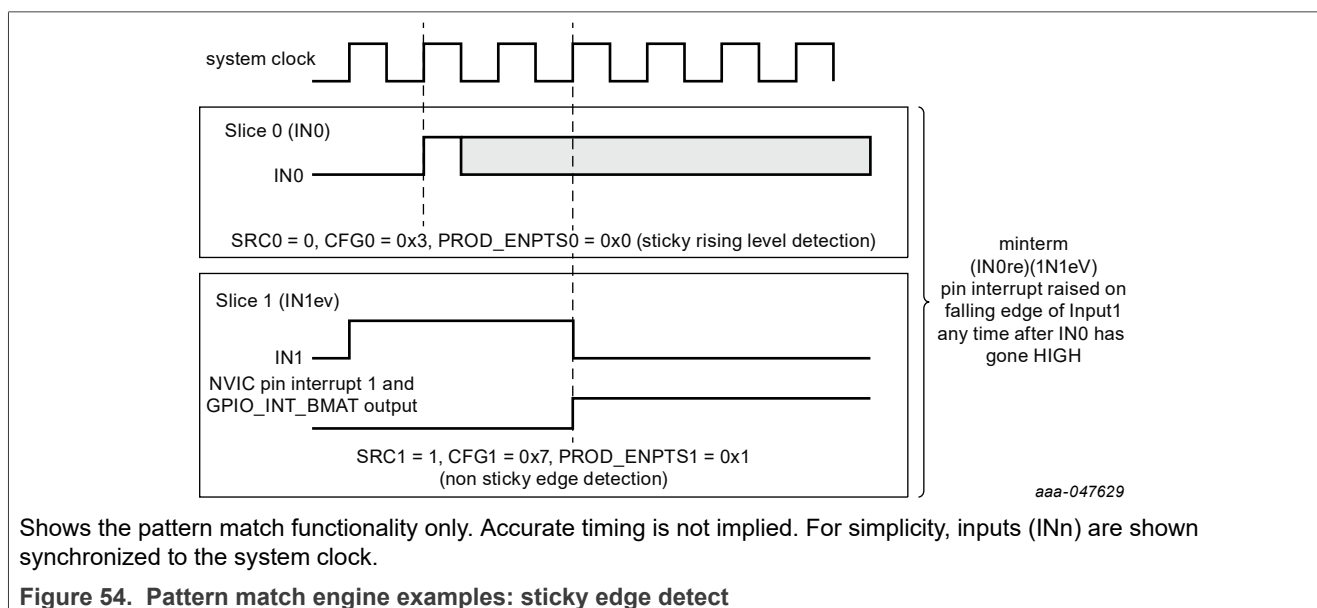
IN7ev = (non-sticky) event (rising or falling edge) on input 7

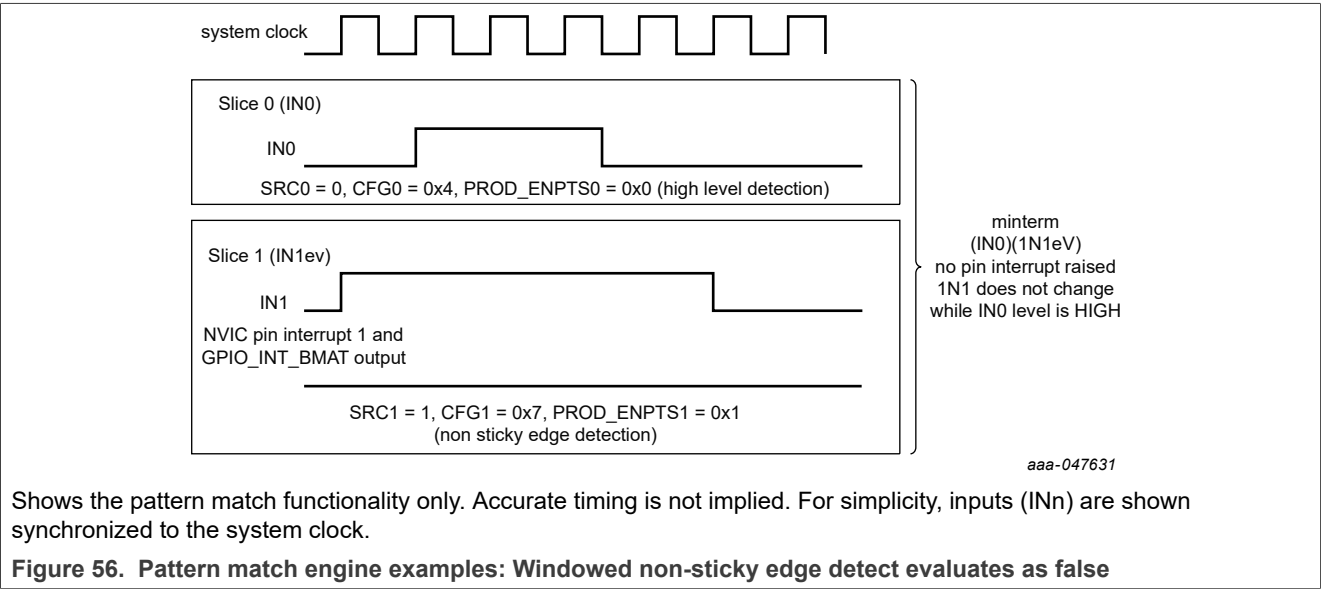
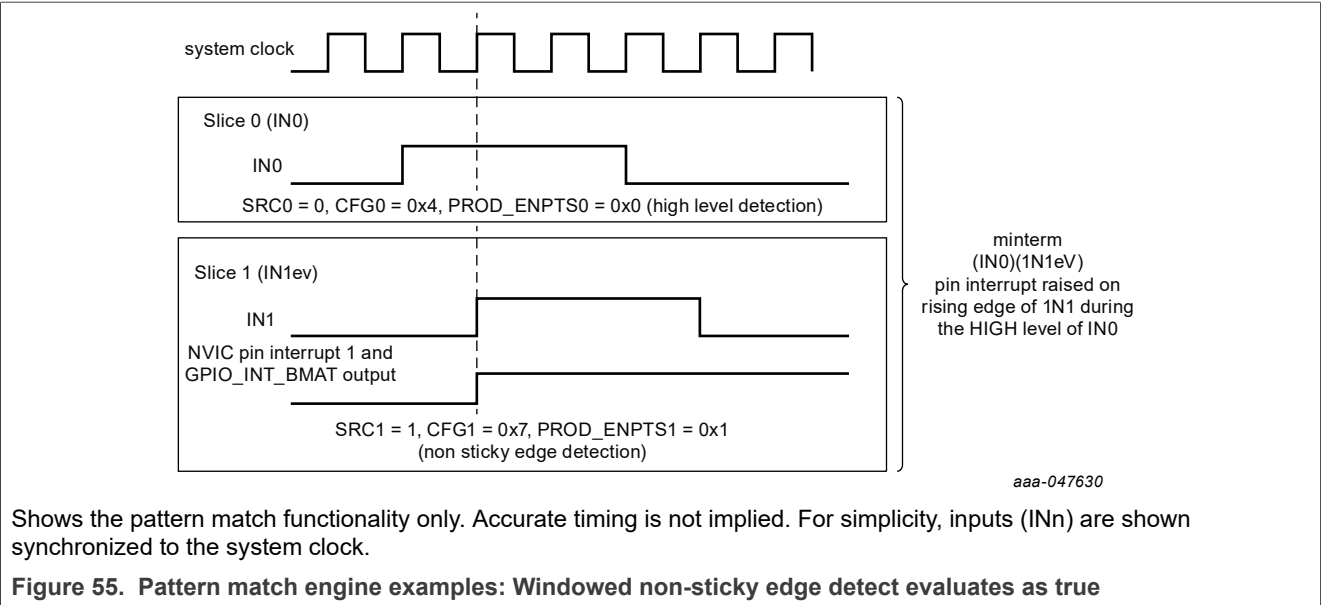
One bit-slice controls each individual term in the expression shown above. To specify this expression, program the pattern-match bit-slice source and configuration register fields as follows:

- PMSRC register, see the section "Pattern match interrupt bit-slice source register (PMSRC)" in the Register description ([Ref. 5](#)).
 - Since bit slice 5 is 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 the section "Pattern match interrupt bit slice configuration register (PMCFG)" in the Register description ([Ref. 5](#)).
 - 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 the section "Pattern match interrupt control register (PMCTRL)" in the Register description ([Ref. 5](#)).
 - Bit0: Setting this bit selects pattern matches to generate the pin interrupts in place of the normal pin interrupt mechanism.
For this example, pin interrupt 0 is 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 is asserted in response to a match on the second product term.
Pin interrupt 5 is asserted when there is a match on the third product term.
Pin interrupt 7 is asserted on a match on the last term.
 - Bit1: When a match occurs on ANY of the product terms in the expression, setting this bit causes the RxEv signal to the CPU to be asserted. Otherwise, the RXEV line is not used.
 - Bit31:24: If the corresponding product terms are currently matching, bits 0, 2, 5 and/or 7 may be high at any given time.
 - The remaining bits are always low.

19.5.3 Pattern match engine edge detect examples





20 Secure pin interrupt and pattern match (SECPINT)

The pin interrupt generator and the pattern match engine are available on all MCX W23 devices. Secure PINT (or GPIO_INT_SEC) uses secure GPIO functions but also any other pin input functions.

20.1 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 edges and/or falling edges.
 - 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 generate an RXEV notification to CPU0.
 - To create complex state machines based on pin inputs, pattern match can be used with software.

20.2 Basic configuration

- Pin interrupts
 - In the input multiplexer block, select up to two external interrupt pins from all digital pins on port 0 (see the register descriptions in the section "Pin interrupt and pattern match (PINT)" of the Register description ([Ref. 5](#)). 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 the section "AHB clock control 2 (AHBCLKCTRL2)" in the Register description ([Ref. 5](#)).
 - 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 the section "Inputmux register descriptions" in the Register description ([Ref. 5](#)). The pin selection process is the same for pin interrupts and the pattern match engine. So, the two features are mutually exclusive.
 - Enable the clock to the pin interrupt register block in the AHBCLKCTRL2 register (see the section "AHB clock control 2 (AHBCLKCTRL2)" in the Register description ([Ref. 5](#)).
 - 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).

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

To configure pins as pin interrupts, follow these steps:

1. Determine the pins that serve as pin interrupts on the MCX W23 package. For determining the GPIO port pin number associated with the package pin, see the data sheet ([Ref. 7](#)).
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.
Note: 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 the P0_SEC(n) function).
3. Enable each pin interrupt in the NVIC.

When the pin interrupts or pattern match inputs are configured, the pin interrupt detection levels or the pattern match boolean expression can be set up. For the PINTSECSEL registers, see the section "Pin interrupt select register (PINTSELa)" in the Register description ([Ref. 5](#)).

Note: 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.

20.3 Pin description

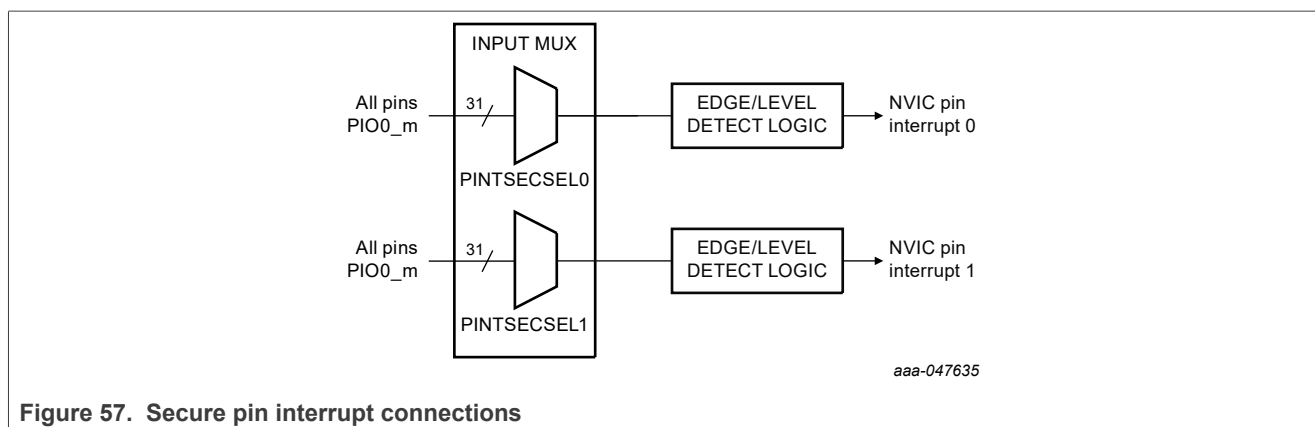
The pin interrupt secure select registers (PINTSECSELn) in the input multiplexer determine the inputs to the pin interrupt and pattern match engine (see the section "Pin interrupt select register (PINTSELa)" in the Register description ([Ref. 5](#)).

20.4 General description

Pins with configurable functions can serve as external interrupts or inputs to the pattern match engine. For these features, up to two pins can be configured using the PINTSECSEL registers in the input multiplexer block.

20.4.1 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 the register descriptions in the section "Pin interrupt and pattern match (PINT)" of the Register description ([Ref. 5](#)). The external interrupt pins are connected to two individual interrupts in the NVIC. They are created based on rising edges or falling edges or on the input level on the pin.



20.4.2 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. When the minterm evaluates as true, a pin interrupt is asserted.

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

- Edge with memory (sticky):
A rising edge and/or a 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 level or LOW level on the selected input.

Figure 59 shows the details of the edge detection logic for each slice in secure PINT block.

To create a pin interrupt when a rising or falling edge occurs after a qualifying edge event, sticky events can be combined with non-sticky events.

A time window can be created during which rising edges or falling edges can create a pin interrupt by combining a level detect with an event detect (for details, see Section 20.5.3).

Figure 58 shows the connections between the pins and the pattern match engine. All pins that are inputs to the pattern match engine can be GPIO port pins or other pin functions, depending on the IOCON configuration.

Note: To operate, the pattern match feature requires clocks. So, it cannot generate an interrupt or wake up the device during deep-sleep mode.

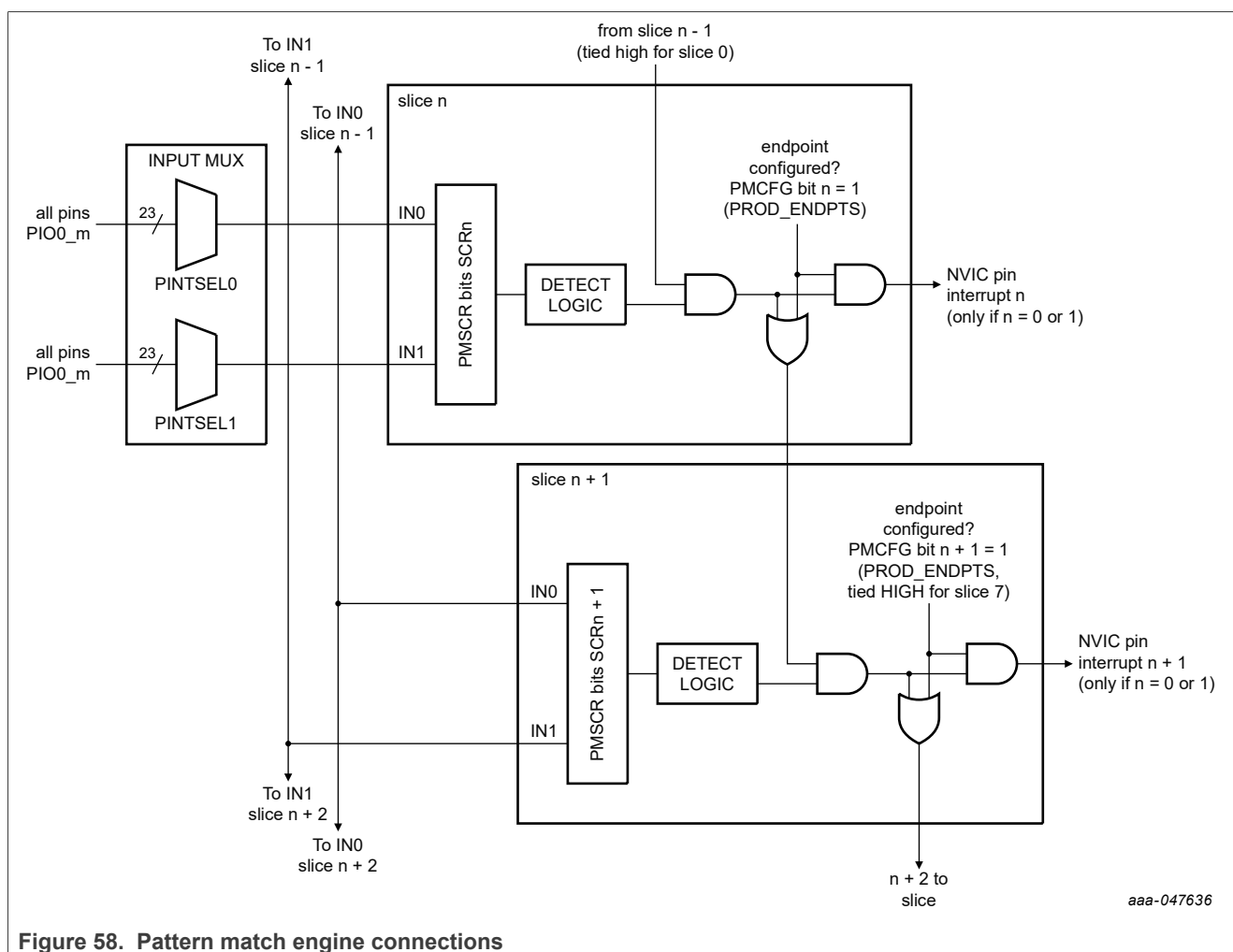


Figure 58. Pattern match engine connections

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 the first minterm, IN0 falling edge AND NOT 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 is asserted.
- In the second minterm, IN0 rising edge AND 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 is asserted but not 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 20.5.2](#)

20.5 Functional description

20.5.1 Pin interrupts

In this interrupt facility, the pin interrupt secure select registers (PINTSECSEL0-1) identify up to 8 pins as interrupt sources. All registers in the pin interrupt block contain 2 bits, corresponding to the pins the PINTSECSEL0-1 registers call. The ISEL register defines if each interrupt pin is edge-sensitive or level-sensitive. The RISE and FALL registers detect edges on each interrupt pin. They can be written to clear (and set) edge detection. The IST register indicates if each interrupt pin is requesting an interrupt. 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 169](#).

Table 169. 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

20.5.2 Pattern match engine example

Suppose the desired boolean pattern to be matched is:

$(IN0) + (IN0 * IN1) + (\sim IN0 * IN1fe) + (IN0 * IN1ev) + (IN0re)$, where:

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. However, it is given as an example.

One bit slice controls each individual term in the expression shown above. To specify this expression, program the pattern-match bit-slice source and configuration register fields as follows:

- PMSRC register, see the section "Pattern match interrupt bit-slice source register (PMSRC)" in the Register description ([Ref. 5](#)):
 - Since bit slice 4 is 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 the section "Pattern match interrupt bit slice configuration register (PMCFG)" in the Register description ([Ref. 5](#)).
 - 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 the section "Pattern match interrupt control register (PMCTRL)" in the Register description ([Ref. 5](#)).
 - Bit0: Setting this bit selects pattern matches to generate the pin interrupts in place of the normal pin interrupt mechanism.
For this example, pin interrupt 0 is asserted when a match is detected on the first product term (which, in this case, is just a high level on input 1).
Pin interrupts 2, 4, 6, and/or 7 are asserted in response to a match on the second, the fourth, the sixth, and/or the seventh product term. However, they are not used since they are not driven to the NVIC.
 - Bit1: Setting this bit causes the RxEv signal to the CPUs to be asserted whenever a match occurs on ANY of the product terms in the expression. Otherwise, the RXEV line is not used.
 - Bit31:24: If the corresponding product terms are currently matching, bits 0, 2, 4, 6 and/or 7 may be high at any given time.
 - The remaining bits are always low.

20.5.3 Pattern match engine edge detect examples

For the examples, see [Section 19.5.3](#).

21 Group GPIO input interrupt (GINT0)

21.1 Features

- To contribute to a combined group interrupt, the inputs from any number of digital pins can be enabled.
- 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.
- Logically combined output can be configured to wakeup upon edge-triggered or level-triggered.
- 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 the GROUP0 register interface in the AHBCLKCTRL0 register: GINT field. For sleep mode, the group interrupt wake-up feature is enabled in the CPUs NVIC. For deep-sleep and power-down low-power modes, the group interrupt wake-up feature is enabled via the relevant low-power API.

The pins can be configured as GPIO pins through IOCON, but they do not 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 GINT0 module can monitor the state of the selected digital input pins and generate an interrupt or wakeup when the specified state is observed.

By programming the configuration registers, the application can determine :

- which input pins must be monitored (enable bit)
- which polarity must be used for each pin (polarity bit to indicate if input signal must be inverted or not)
- if all pins must have the required state (logical AND of all enabled pins), or if any of the pins matches the required state (logical OR of all enabled pins)
- if an interrupt is generated when either a high level or a rising edge is detected on the output of the logical AND or logical OR.

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 asynchronously wakes up the part before asserting the interrupt request.

Writing a one to the interrupt status bit in the control register clears the interrupt request line.

21.4 Functional description

To contribute to a common group interrupt (GINT), any subset of the pins in a port can be selected. They can be enabled to wake up the part from deep-sleep and power-down modes.

An interrupt can be requested based on any selected subset of pins within the port. The pins that contribute to the port interrupt are selected by 1s in the enable register of the port.

A polarity can be selected for each pin in the polarity register of the port (as-is or inverted). The level on each pin is put together with its polarity bit into an XOR gate. The output of this XOR gate is put together with its enable bit into an AND gate.

The outputs of all these AND gates are fed into an AND gate and an OR gate. The application can select either the output of the AND gate or the output of the OR gate to generate the interrupt request level of the port. The application can decide to either use the high level of this signal to generate an interrupt and/or wakeup, or to feed this signal into a rising edge detection block which sets the interrupt and/or wakeup source.

The raw interrupt request from the group interrupt is sent to the NVIC (see [Table 8](#)).

Example : Wakeup upon falling edge of a single input pin

If the application wants to wakeup from a rising edge on the input pin PIO0_0, it must program the GINT registers with following values :

- PORT_ENA register to 0x1 (Enable PIO0_0 pin only)
- PORT_POL register to 0x1 (Maintain polarity of input signal)
- CTRL register to 0x0 (Edge-triggered interrupt)

22 DMA controller (DMA0 - DMA1)

The DMA controllers are available on all MCX W23 devices.

22.1 Features

- The DMA controller has two instances. DMA0 and DMA1. The user can decide for each one if it is intended for nonsecure use or secure use.
- DMA0:
 - 8 channels, with multiplexers for 21 trigger sources. 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 DMA channel output triggers. SHA-2 and AES also provide DMA channel and trigger interfaces.
- DMA1:
 - 8 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.

22.2 Basic configuration

Configure the DMA as follows:

- To enable the clock to the DMA0 registers interface, use the AHBCLKCTRL0 register (see the section "AHB clock control 0 (AHBCLKCTRL0)" in the Register description ([Ref. 5](#))).
- To enable the clock to the DMA1 registers interface, use the AHBCLKCTRL2 register (see the section "AHB clock control 2 (AHBCLKCTRL2)" in the Register description ([Ref. 5](#))).
- Clear the DMA0 peripheral reset using the PRESETCTRL0 register (see the section "Peripheral reset control 0 (PRESETCTRL0)" in the Register description ([Ref. 5](#))).
- Clear the DMA1 peripheral reset using the PRESETCTRL2 register (see the section "Peripheral reset control 2 (PRESETCTRL2)" in the Register description ([Ref. 5](#))).
- The DMA controller provides an interrupt to the NVIC (see [Section 3](#)).
- Most peripherals that support DMA have at least one DMA request line associated with them. The related channels must be set up according to the desired operation. DMA requests and triggers are described in detail in [Section 22.4.1](#).
- For peripherals using DMA requests, DMA operation must be triggered before any transfer occurs. It can be done using software. Optionally, one of the hardware triggers can signal it through the input multiplexers registers DMA_ITRIG_INMUX[0:22]. DMA requests and triggers are described in detail in [Section 22.4.1](#).
- Trigger outputs may 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 18.4.4](#).

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 selected timers and pin interrupts may also be used as DMA triggers. In addition, four DMA triggers can be selected from among all DMA channel output triggers. SHA-2 and AES also provide DMA channel and trigger interfaces.

If the active security profile enables it, the DMA controller can access all on-chip RAM and flash memories, all AHB peripherals (potentially excepting the ones that do not support DMA or contain their own DMA engine), and all APB peripherals.

If TZ-enabled devices to support DMA operation for secure thread and nonsecure thread are used, two DMAs are available on this device. One DMA is available for nonsecure mode. The bootloader code in ROM (using the settings for Trustzone/AHB secure controller defined by the application) can program the other DMA to serve as the secure DMA. The number of available channels and trigger MUXs available on the second DMA is reduced. Also, to protect secure DMA from nonsecure requests or triggers, a special masking mechanism is implemented that enables DMA access for a particular device only if the mask is securely disabled.

22.3 Pin description

The DMA controller has no direct pin connections. However, some DMA triggers can be associated with pin functions (see [Section 22.4.1.2](#)).

22.4 General description

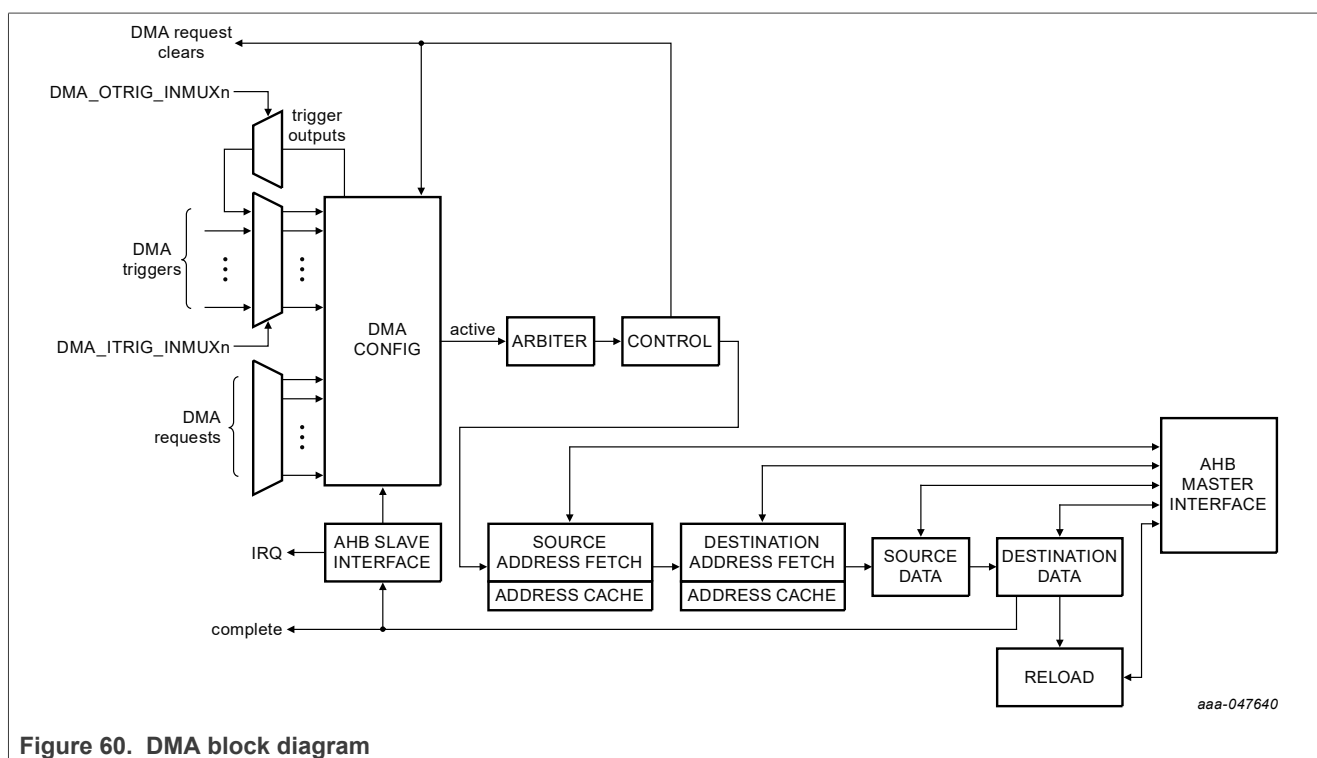


Figure 60. DMA block diagram

22.4.1 DMA requests and triggers

In general, DMA requests are intended to pace transfers to match what the peripheral (including its FIFO if it has one) can do. For example, the USART issues a transmit DMA request when its transmit FIFO is not full. It issues a receive DMA request when its receive FIFO is not empty. DMA requests are summarized in [Table 170](#).

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-related or pin-related events occur. If a peripheral is involved in the transfer, a peripheral DMA request paces these transfers.

Note: No DMA activity takes place for any particular DMA channel unless either software or hardware triggers that channel. DMA triggers are summarized in [Table 170](#).

When software or hardware does trigger a DMA channel, a DMA request, if it is enabled for that channel, initiates a DMA operation on that specific channel.

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

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.

22.4.1.1 DMA requests

DMA requests are directly connected to the peripherals. Each channel supports one DMA request line and one trigger input. Some DMA requests allow a selection of requests sources. DMA triggers are selected from many possible input sources. [Table 170](#) and [Table 171](#) give the requests and trigger MUXs for DMA controller 0 and 1n.

Table 170. 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	spare channel, no request connected	DMA0_ITRIG_INMUX2
3	SPIFI	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 2 RX/I2C slave ^[1]	DMA0_ITRIG_INMUX8
9	Flexcomm interface 2 TX/I2C master ^[1]	DMA0_ITRIG_INMUX9

[1] See section below for information about DMA for the I²C monitor function.

Table 171. DMA1 requests and trigger multiplexers

SDMA 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	spare channel, no request connected	DMA1_ITRIG_INMUX2
3	SPIFI	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 2 RX/I2C slave ^[1]	DMA1_ITRIG_INMUX8

Table 171. DMA1 requests and trigger multiplexers...continued

SDMA channel	Request input	DMA trigger mux
9	Flexcomm interface 2 TX/I2C master ^[1]	DMA1_ITRIG_INMUX9

[1] See section below for information about DMA for the I²C monitor function.

DMA with I2C monitor mode

If one of the channels related to the same Flexcomm interface is available, the I²C monitor function may be used with DMA.

Table 172. DMA with the I2C

I2C master DMA	I2C slave DMA	I2C monitor DMA
not enabled	-	If the I2C monitor DMA is enabled, it uses the DMA channel for the master function of the same Flexcomm interface.
enabled	not enabled	If the I2C monitor is DMA enabled, it uses the DMA channel for the slave function of the same Flexcomm interface.
enabled	enabled	The I2C monitor function cannot use DMA.

22.4.1.2 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 21 possible internal trigger sources for each DMA channel on DMA controller 0 and 15 possibilities for SDMA 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 173](#) and the section "Input multiplexing (INPUT MUX)" in the Register description ([Ref. 5](#)).

Table 173. 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
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	n/a	hash-crypt output DMA

Table 173. DMA trigger sources...continued

DMA trigger	DMA0 trigger input	DMA1 trigger input
15	DMA output trigger 0	n/a
16	DMA output trigger 1	n/a
17	DMA output trigger 2	n/a
18	DMA output trigger 3	n/a
19	SCT0 DMA request 0	n/a
20	SCT0 DMA request 1	n/a
21	hash-crypt output DMA	n/a

22.4.1.3 Trigger operational detail

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

If a channel is configured with the SWTRIG bit equal to 0, hardware or software can trigger the channel later. Software triggering is accomplished with 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.

When triggered, DMA requests pace the transfer on a channel if the PERIPHREQEN bit in the related CFG register is set. Otherwise, the transfer proceeds at full speed.

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

22.4.1.4 Trigger output detail

Each channel of the DMA controller provides a trigger output. To support complex transfers on selected peripherals, it allows the possibility of using the trigger outputs as a trigger source to a different channel. This kind of transfer can, for example, use more than one peripheral DMA request. An example use is to input data to a holding buffer from one peripheral and then output the data to another peripheral, when the appropriate peripheral DMA request paces both transfers. This kind of operation is called chained operation or channel chaining.

22.4.2 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.

When the DMA controller is set up for operation, using any specific DMA channel requires initializing the registers associated with that channel in [Section 22.4.1](#). The channel descriptor must also be programmed, which is located somewhere in memory, typically in on-chip SRAM (see the section "SRAM address of the channel configuration table (SRAMBASE)" in the Register description ([Ref. 5](#)). [Table 174](#) gives the channel descriptor.

Table 174. Channel descriptor

Offset	Description
+ 0x0	reserved

Table 174. Channel descriptor...continued

Offset	Description
+ 0x4	source data end address
+ 0x8	destination end address
+ 0xC	link to next descriptor

The source and destination end addresses and 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 (which 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 (which corresponds to the memory buffer):

Buffer ending address = buffer starting address + (XFERCOUNT * the transfer increment)

For the description of SRCINC and DSTINC, see the section "Transfer configuration register for DMA channel n (XFERCFGn)" in the Register description ([Ref. 5](#)).

Note: *XFERCOUNT is defined as the actual count minus 1, which is why it is not necessary to subtract 1 from the count in the equation above.*

After the channel has enough DMA requests and/or triggers, depending on its configuration, the initial descriptor is exhausted. At that point, if the transfer configuration directs it, the channel descriptor is reloaded with data from memory that the "Link to next descriptor" entry of the initial channel descriptor points. Descriptors loaded in this manner look slightly different from the channel descriptor (see [Table 175](#)). The difference is that a new transfer configuration is specified in the reload descriptor and not written to the XFERCFG register for that channel.

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

Table 175. Reload descriptors

Offset	Description
+ 0x0	transfer configuration
+ 0x4	source end address. 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. 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 (that is, the size of a descriptor).

22.4.3 Single buffer

The single buffer applies to memory-to-memory moves and peripheral DMA that occurs only occasionally and is set up for each transfer. For this kind of operation, only the initial channel descriptor shown in [Table 176](#) is necessary.

Table 176. Channel descriptor for a single transfer

Offset	Description
+ 0x0	reserved
+ 0x4	source data end address
+ 0x8	destination end address
+ 0xC	not used

When the reload bit field in the XFERCFG register is programmed to 0, this case occurs. When the DMA channel receives a DMA request or trigger (depending on how it is configured), it performs one or more transfers as configured and then stops. When the channel descriptor is exhausted, additional DMA requests or triggers have no effect until software updates the channel configuration.

22.4.4 Ping-pong

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

A Ping-pong transfer uses two buffers alternately. At any one time, DMA operations load or unload one buffer. Software handles the opposite operation, readying the buffer for use when the buffer the DMA controller uses is full or empty. [Table 177](#) shows an example of descriptors for ping-pong from a peripheral to two buffers in memory.

Table 177. 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 specify reload. When the channel descriptor is exhausted, descriptor B is loaded using the link to descriptor B. 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. When descriptor A is exhausted, the process repeats, alternately using each of the two memory buffers.

22.4.5 Interleaved transfers

One use for the SRCINC and DSTINC configurations (located in the channel transfer configuration registers, XFRCFGn) is to handle data in a buffer such that it is interleaved with other data.

For example, if four data samples from several peripherals must be interleaved into a single data structure, it may be done while the DMA reads in the data. Setting SRCINC to 4x width for each channel involved allows room for four samples in a row in the buffer memory. The DMA places data for each successive value at the next location for that peripheral.

The reverse of this process can be done using DSTINC to deinterleave combined data from the buffer and send it to several peripherals or locations.

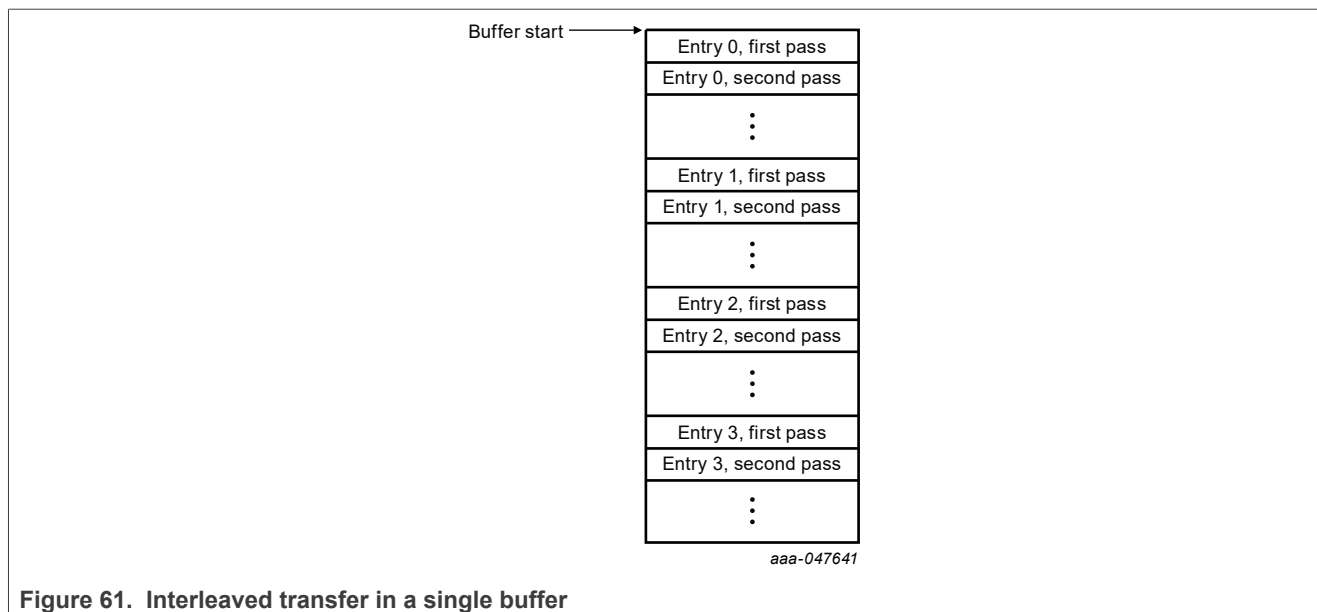


Figure 61. Interleaved transfer in a single buffer

22.4.6 Linked transfers (linked list)

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

An example of a linked transfer can start out like the example for a ping-pong transfer (see [Table 177](#)). The difference is that descriptor B does not link back to descriptor A, but continues on to another descriptor. It can continue as long as required. To repeat a sequence of descriptors, it can be ended anywhere or linked back to any point. However, software can also alter any descriptor not in use.

22.4.7 Address alignment for data transfer

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

22.4.8 Channel chaining

Channel chaining is a feature which allows the completion of a DMA transfer on channel x to trigger a DMA transfer on channel y. This feature can, for example, be used to have DMA channel x read n bytes from UART to memory. And then have DMA channel y transfer 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 is 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), enable 'clear trigger on descriptor exhausted' by setting bit CLRTRIG in the transfer configuration of the channel 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.

After completion of channel x, the descriptor may be reloaded (if configured so) but remains untriggered. To configure the chain to autotrigger itself, set up channels x and y for channel chaining as described above. In addition to setting up channels x and y for channel chaining:

- A ping-pong configuration for channel x and y is recommended, so channel x does not alter data channel y is moving.
- For channel x:
 - To use the same DMA trigger multiplexer as chosen for channel y, configure the input trigger input multiplexer register (DMA_ITRIG_INMUX[0:21]) 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.

22.4.8.1 DMA in reduced power mode

In sleep mode, the DMA can operate and access all enabled SRAM blocks, without waking up the CPU.

23 SCTimer/PWM (SCT)

The SCTimer/PWM is available on all MCX W23 devices.

Note: For a detailed description of SCTimer/PWM applications and code examples, see the application note AN11538 ([Ref. 2](#)).

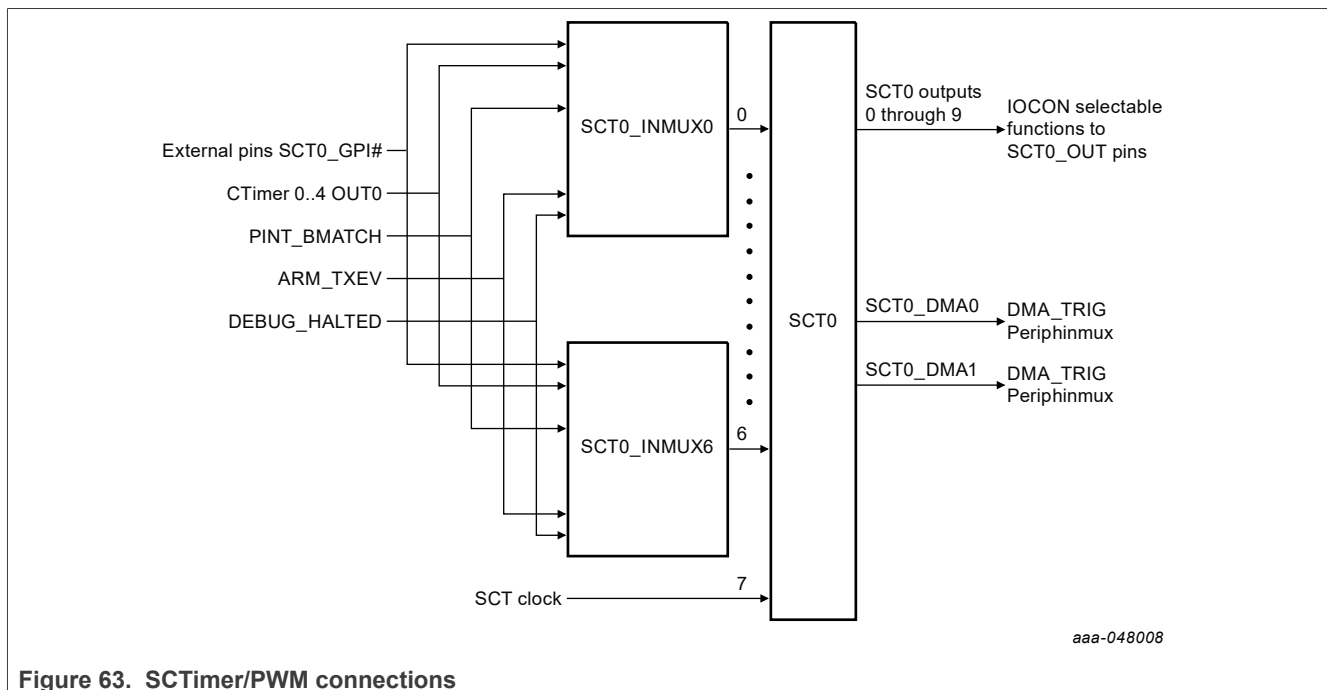
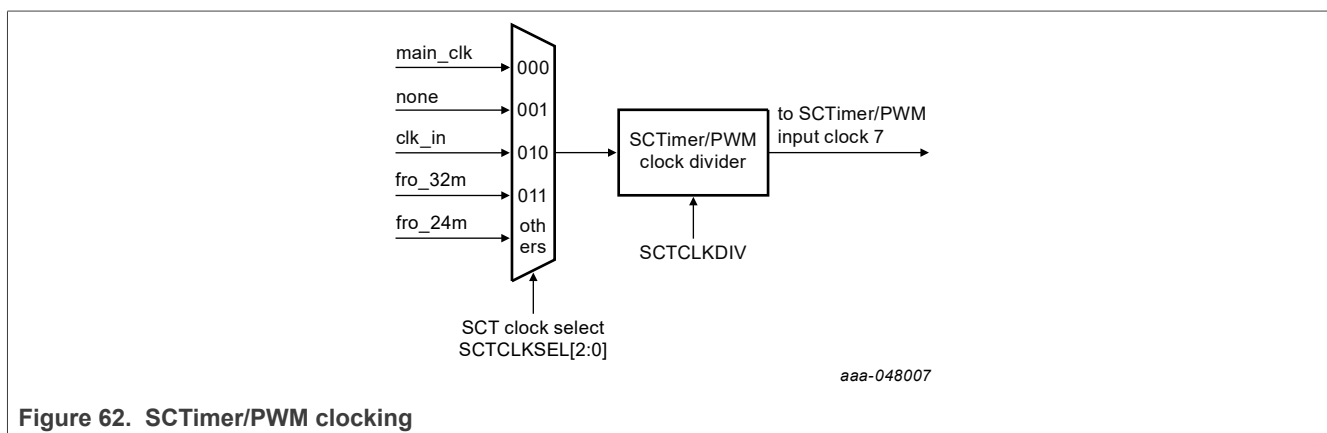
23.1 Features

- The SCTimer/PWM supports:
 - 8 inputs
 - 10 outputs
 - 16 match/capture registers
 - 16 events
 - 32 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 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 cycles and common PWM cycle lengths.
- 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. States (contexts) qualify event creation.
 - 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 predefined meaning. The user determines the states. They provide a mechanism for context switching for the SCT, including creation of complex state machines.
 - The only function of a state 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.

23.2 Basic configuration

Configure the SCT as follows:

- Enable the clock to the SCTimer/PWM (SCT) in the AHBCLKCTRL1 register (see the section "AHB clock control 1 (AHBCLKCTRL1)" in the Register description ([Ref. 5](#)).
- Clear the SCT peripheral reset using the PRESETCTRL register (see the section "Peripheral reset control 1 (PRESETCTRL1)" in the Register description ([Ref. 5](#)).
- The SCT provides an interrupt to the NVIC (see [Section 3](#)).
- SCT inputs are selected from the SCT input multiplexer registers (see [Section 18](#)).
- The SCT DMA request lines are connected to the DMA trigger inputs via the DMA_ITRIG_PINMUX registers (see [Section 18.4.4](#)).



23.3 Pin description

Note: Availability of inputs or outputs related to a particular peripheral function may be package-dependent.

To assign the SCT functions to external pins, see [Section 15.3.1](#).

SCT inputs are selected from the SCT input multiplexer registers (see [Section 18](#)).

SCT outputs can be routed to pins or DMA triggers (see [Section 18](#)).

Table 178. SCT0 pin description (inputs)

Type	Function	Connect to	Use register	Reference
external from pin	SCT0_GPI0	PIO0_19	IOCON register for the related pin	see Section 15
	SCT0_GPI1	PIO0_18		
	SCT0_GPI2	PIO0_17		
	SCT0_GPI3	PIO0_16		
	SCT0_GPI4	PIO0_11		
	SCT0_GPI5	PIO0_10		
	SCT0_GPI6	PIO0_9		
	SCT0_GPI7	PIO0_8		
internal	-	CTIMER0_MAT0 CTIMER1_MAT0 CTIMER2_MAT0 CTIMER3_MAT0 CTIMER4_MAT0	CTIMER register	see Table 184
	-	PINT_BMATCH	PINT register for pattern match engine	see Section 19.4.2
	Arm send event	ARM_TXEV	-	-
	Arm debug	DEBUG_HALTED	-	-

Table 179. SCT0 pin description (outputs)

Type	Function	Connect to	Use register	Reference
external to pin	SCT0_OUT0	PIO0_15	IOCON register for the related pin	see Section 15
	SCT0_OUT1	PIO0_14		
	SCT0_OUT2	PIO0_13		
	SCT0_OUT3	PIO0_12		
	SCT0_OUT4	PIO0_7		
	SCT0_OUT5	PIO0_6		
	SCT0_OUT6	PIO0_5		
	SCT0_OUT7	PIO0_4		
	SCT0_OUT8	PIO0_3		
	SCT0_OUT9	PIO0_2		
internal	-	SDMA trigger	SCT_DMA0, SCT_DMA1	see Section 22.4.1.2

Table 180. Suggested SCT input pin settings

IOCON bits	Type D pin	Type I2C pin
15	not used; set to 0	I2C filter; set to 0 for fast mode; 1 for high-speed mode
14	not used; set to 0	I2C mode; set to 0
13	not used; set to 0	I2C pull-up current source; set to 0 for open-drain mode
12	not used; set to 0	I2C 3 ns glitch filter; set to 0
9	OD; set to 0	not used; set to 0
8	DIGIMODE; set to 1	same as type D
7	INVERT; set to 0	same as type D
6	not used; set to 0	I2C slew rate; set to 1
5:4	MODE; set to 0 (pull-down/pull-up resistor not enabled). If the input is floating sometimes (causing leakage within the pin input), it can be another setting.	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

Table 180. Suggested SCT input pin settings...continued

IOCON bits	Type D pin	Type I2C pin
general comment	good choice for an SCT input	reasonable choice for an SCT input

Recommended IOCON settings are shown in [Table 180](#) and [Table 181](#).

Table 181. Suggested SCT output pin settings

IOCON bits	Type D pin	Type I2C pin
15	not used; set to 0	I2C filter; set to 0 for fast mode; 1 for high-speed mode
14	not used; set to 0	I2C mode; set to 0
13	not used; set to 0	I2C pull-up current source; set to 0 for open-drain mode
12	not used; set to 0	I2C 3 ns glitch filter; set to 0
9	OD; set to 0	not used; set to 0
8	DIGIMODE; set to 1	same as type D
7	INVERT; set to 0	same as type D
6	standard slew; set to 0	I2C slew rate; set to 1
5:4	MODE; set to 0	not used; set to 0.
3:0	FUNC Must select the correct function for this peripheral.	same as type D
general comment	Good choice for an SCT output.	Not recommended for SCT outputs.

23.4 General description

The SCTimer/PWM is a powerful, flexible timer module that can create complex PWM waveforms. It can also perform 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 mode or bidirectional mode. As with most timers, the SCT supports a selection of match registers to which the count value can be compared. It also supports capture registers where the current count value can be recorded when a predefined condition is detected.

An additional feature contributing to the versatility of the SCT is the concept of "events". The SCT module supports multiple separate events that the user can define based on a combination of parameters. The combination may include 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, 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). Multiple states with different sets of enabled and disabled events are allowed. Changing from one state to another is event-driven as well. So, it can happen without software intervention. Any event can dictate to remain in the current state or switch to a new one. By defining these states, the SCTimer/PWM provides the means to alter the entire behavior of the machine periodically, based on whatever criteria the user chooses. To accomplish complex waveform and timing tasks, it is also possible to generate finite state machines in hardware with any desired level of complexity.

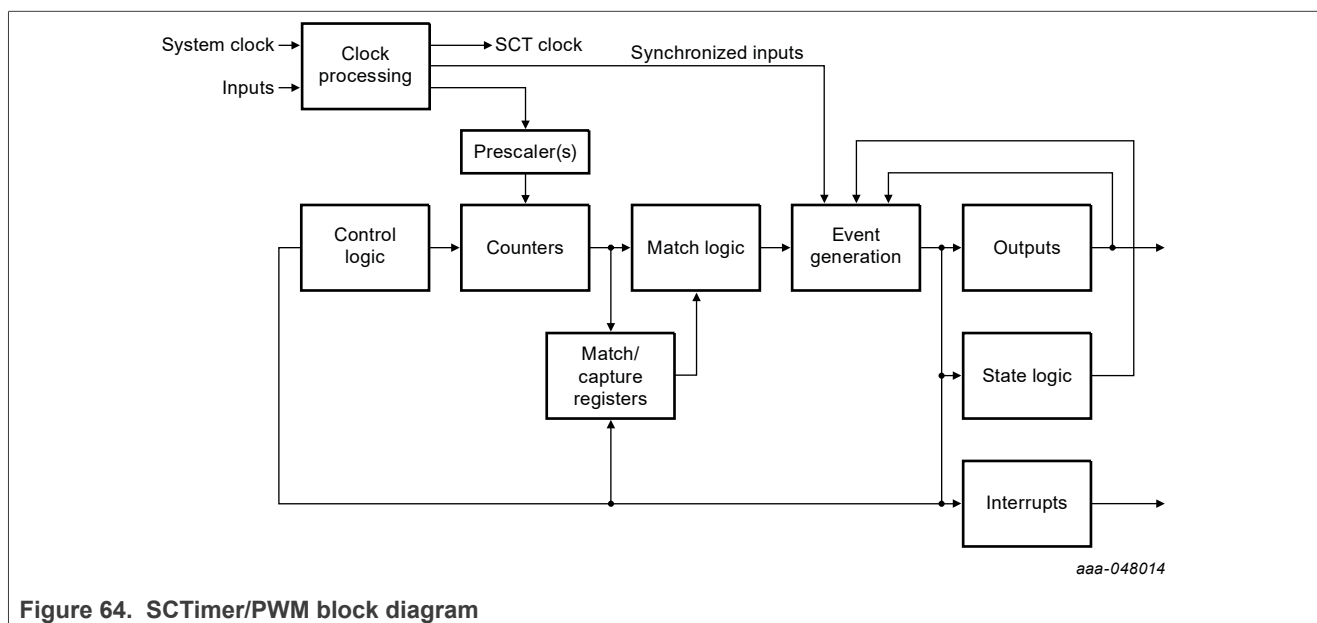
In a simple system, such as a basic timer/counter with capture and match capabilities, using more than a single state is not required. All events that can cause the timer to capture the timer value or toggle a match output are always enabled while the counter is running. In this case, no events are filtered and a single state that does not change describes the system. This setup is the default configuration of the SCT.

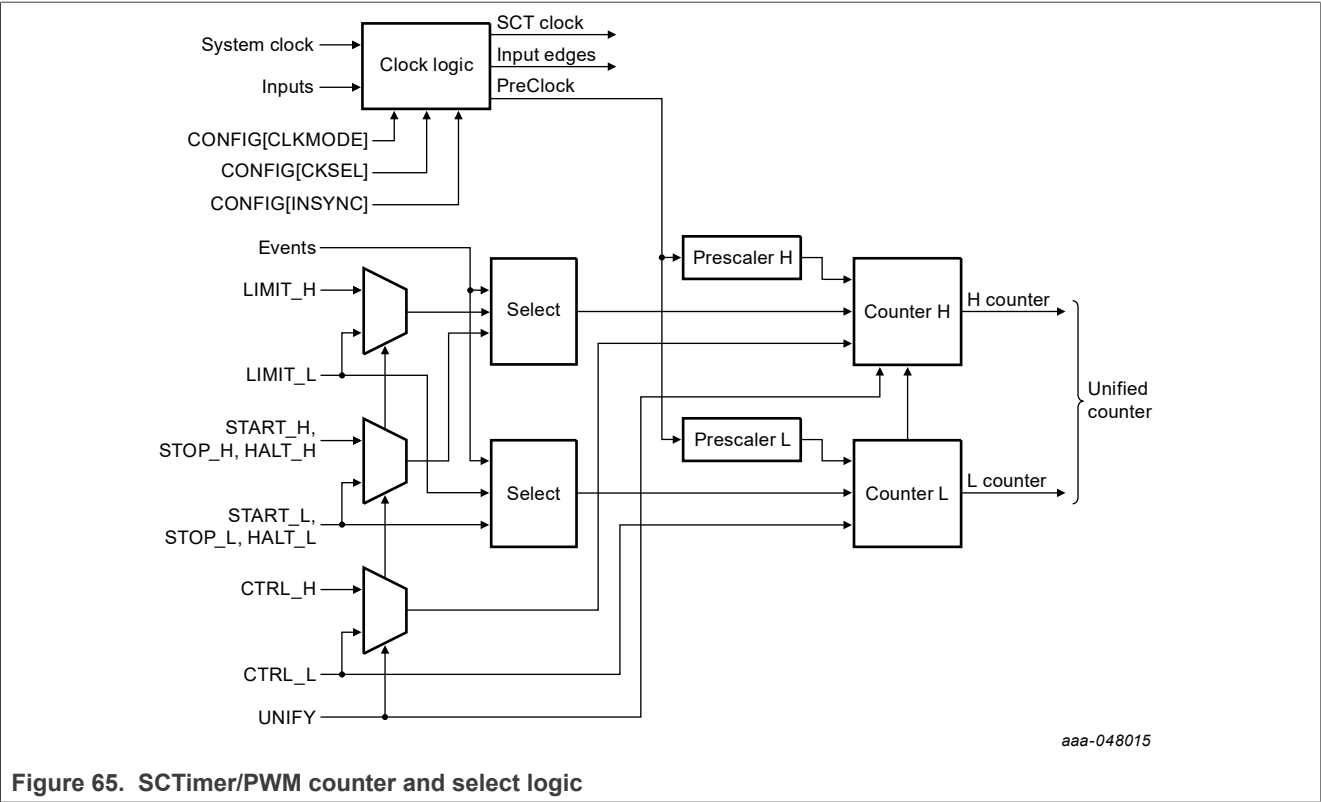
In a more complex system, two states can be set up that allow certain events in one state and not in the other. An event enabled in both states can then be used to move from one state to the other and back while filtering out other events in either state. In such a two-state system, different waveforms at the SCT output can be created, depending on the event history. Changing between states is event-driven and happens without any CPU intervention.

For more advanced applications, up to 32 different states/contexts can be defined (depending on the number of states available on a particular part). If necessary, 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 gives the SCT the unique capability to be used 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 other timers, and so on) 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 event generation

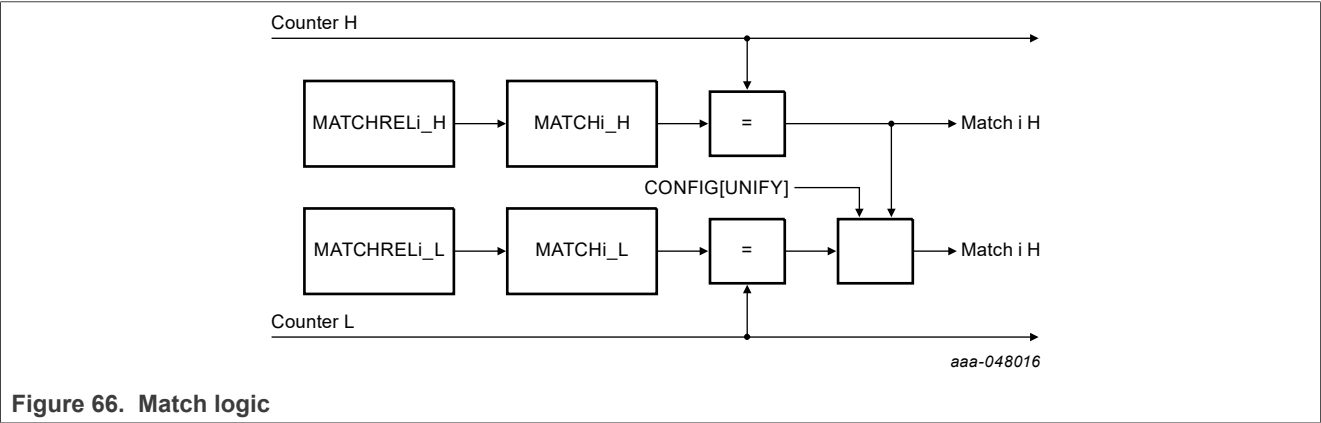




Note: In this section, the term bus error indicates an SCT response that makes the processor take an exception.

23.5 Functional description

23.5.1 Match logic



23.5.2 Capture logic

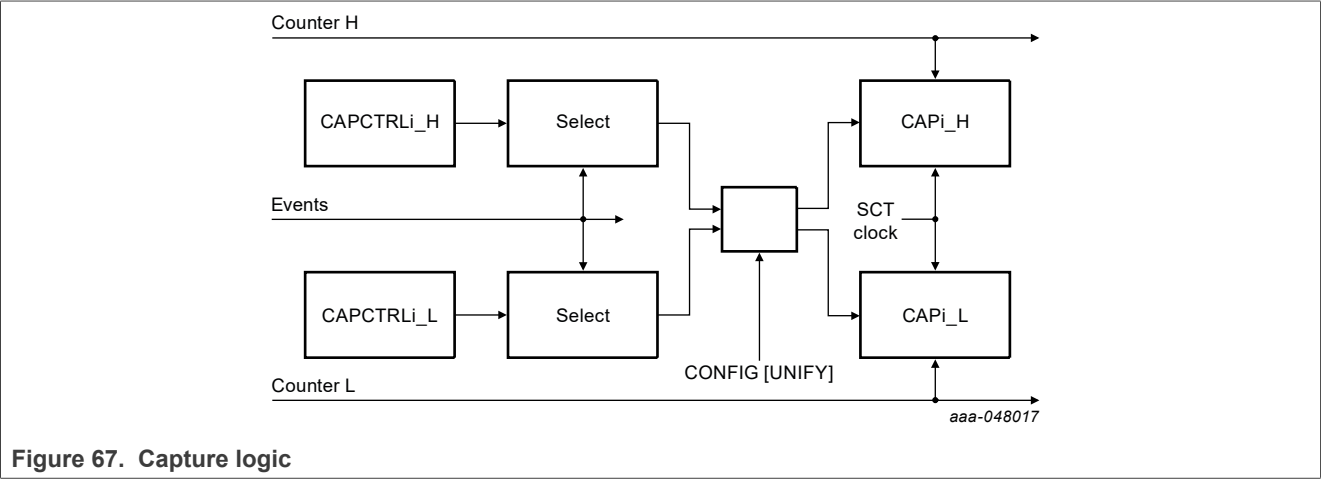


Figure 67. Capture logic

23.5.3 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.

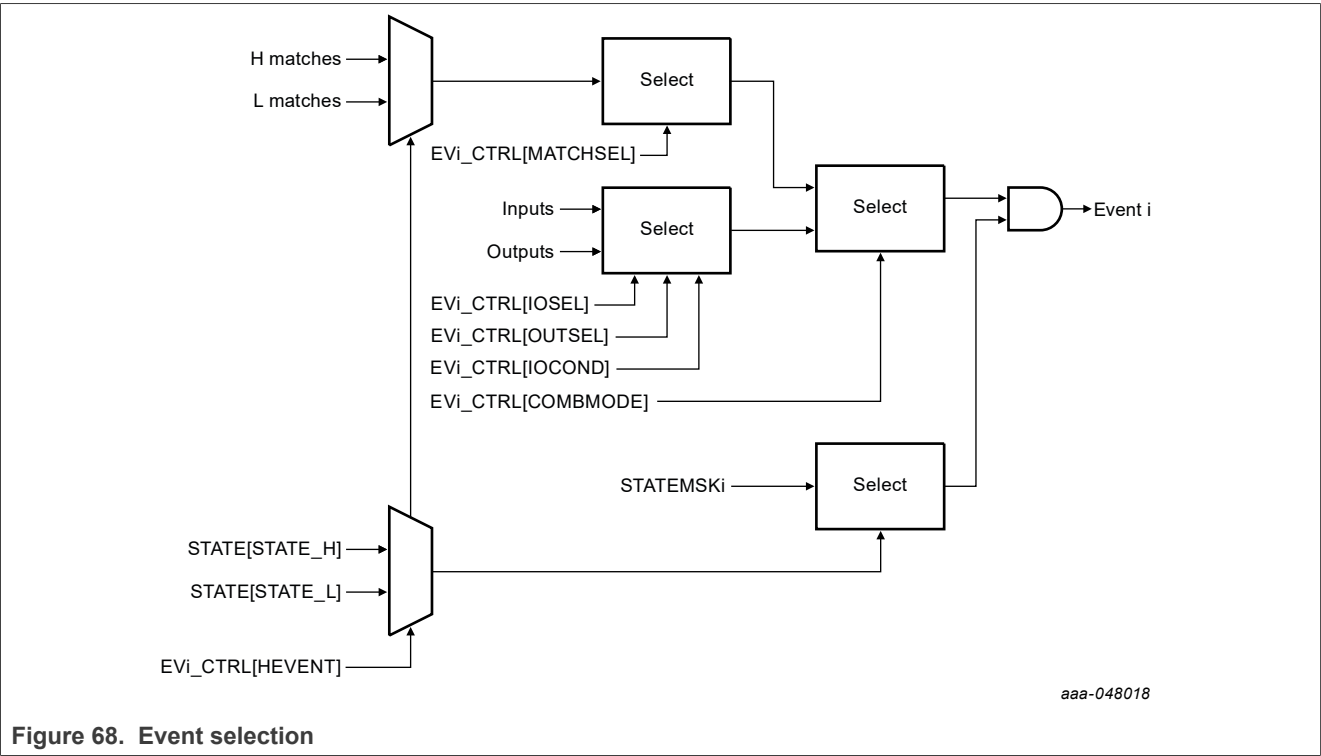
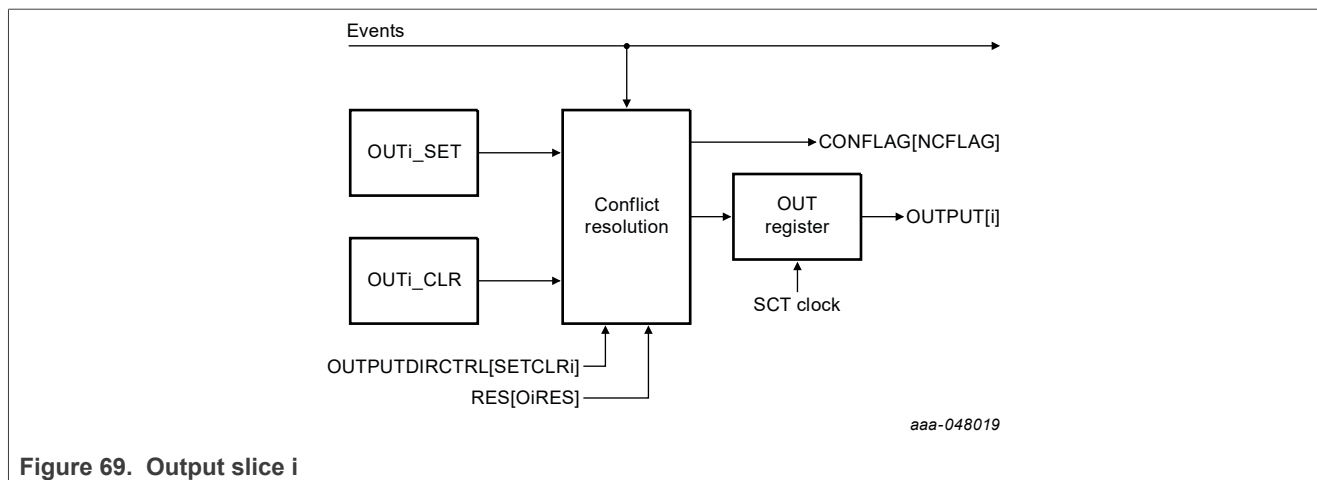


Figure 68. Event selection

23.5.4 Output generation

Figure 69 shows one output slice of the SCT.



23.5.5 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 multistate 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 the EV_CTRL register of each event in one of the following ways:

- The event can increment the current state number with 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.

When the event moving the SCT into a locked state occurs, software can capture the counter value (and potentially create an interrupt and write to all outputs). Later, while the SCT is in the locked state, software can read the counter again to record the time passed since the locking event. To obtain the current state number, software can also read the state variable.

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 with a different value.

23.5.6 Interrupt generation

The SCT generates one interrupt to the NVIC.

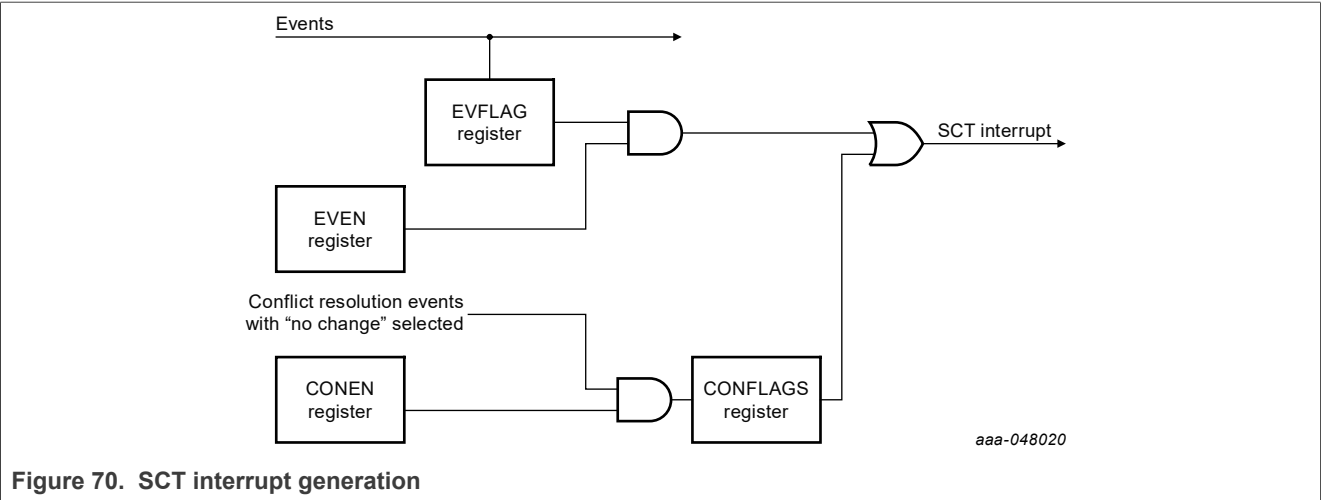


Figure 70. SCT interrupt generation

23.5.7 Clearing the prescaler

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

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

When BIDIR is 0, a limit event caused by an I/O signal can clear a non-zero prescaler. However, a limit event caused by a match-only clears a non-zero prescaler in one special case as described in [Section 23.5.8](#).

When BIDIR is 1, a limit event does not clear the prescaler. If the counter is enabled in that clock, it clears the DOWN bit in the control register and decrements the counter on the same clock.

23.5.8 Match versus I/O events

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

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

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 182](#) shows when the various kinds of events can occur.

Table 182. Event conditions

COMBMODE	IOMODE	Event can occur on clock:
IO	any	Event can occur when HALT = 0 (type A).
MATCH	any	Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (type C).

Table 182. Event conditions...continued

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

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

In a multistate configuration, states change in response to events. When the event occurs, a state change is an additional action that the SCT can perform. When an event to change the state is configured, 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 in this way create a large variety of event-controlled transitions on the SCT outputs and/or interrupts.

When configured, the SCT can run continuously without software intervention. It can generate multiple output patterns under the control of events.

- To configure the SCT, see [Section 23.5.10](#).
- To start, run, and stop the SCT, see [Section 23.5.11](#).
- To configure the SCT as simple event controlled counter/timer, see [Section 23.5.12](#).

23.5.10 Configure the SCT

The sections below show the configuration steps to set up the SCT for multiple events and states.

23.5.10.1 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.

23.5.10.2 Configure the match and capture registers

1. Select how many match and capture registers that 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 if the register is used as a match register or a capture register.
2. Define match conditions for each match register selected:
 - If a 32-bit counter is used, each match register MATCH sets one match value. If the L and H 16-bit counters are used, each match register MATCH sets two match values.

- 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.

23.5.10.3 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 if the event occurs on an input or an 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 if 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 an input changing defines, can be used to restart the counter.
4. Define which events contribute to the SCT interrupt:
 - To enable the event to contribute to the SCT interrupt, set the corresponding event bit in the EVEN and the EVFLAG registers.

23.5.10.4 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.
Note: 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.

23.5.10.5 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, writing to the OUTPUTDIRCTRL register can set the effect of the set and clear of an output to depend on if the counter is counting up or down.

23.5.11 Run the SCT

1. Configure the SCT, see [Section 23.5.10](#).
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.
Note: The counter starts counting once the STOP bit is cleared as well. If the STOP bit is set, the SCT waits for an event to occur that is configured to start the counter.
 - For each counter, select unidirectional or bidirectional counting mode (field BIDIR_L and/or BIDIR_H).
 - Select the prescale factor for the counter clock (CTRL register).
 - Clear the HALT_L and/or HALT_H bit. By default, the counters are halted and no events can occur.
4. To stop the counters using 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, an event configured to clear the STOP bit or software writing a zero to the STOP bit can start the counter again.
 - When the counters 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 using software (independent 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.

23.5.12 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.
The event is allowed to occur in a single state which never changes while the counter is running.

23.5.13 SCT PWM example

[Figure 71](#) 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.

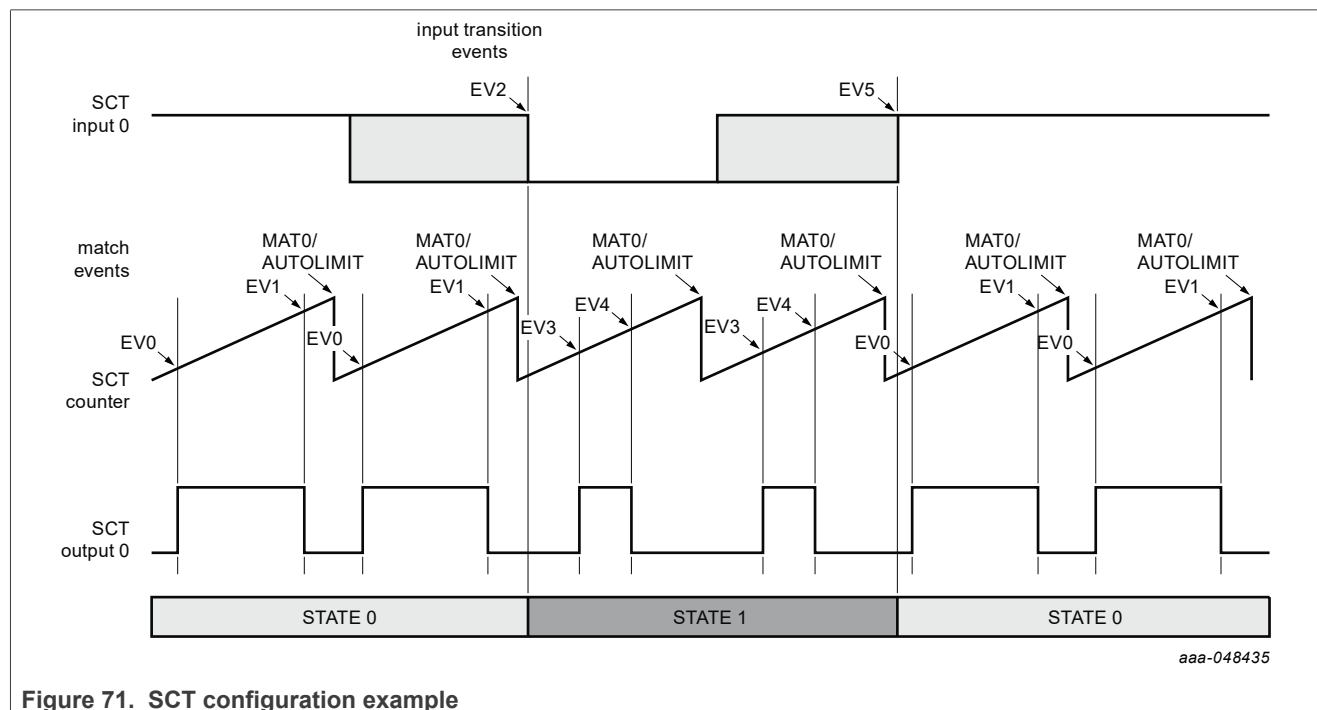


Figure 71. SCT configuration example

This application of the SCT uses the following configuration (all register values not listed in [Table 183](#) are set to their default values):

Table 183. SCT configuration example

Configuration	Registers	Setting
Counter	CONFIG	Uses one counter (UNIFY = 1).
	CONFIG	Enable the autolimit for MAT0. (AUTOLIMIT = 1.)
	CTRL	Uses Unidirectional counter (BIDIR_L = 0).
Clock base	CONFIG	Uses default values for clock configuration.
Match/Capture registers	REGMODE	Configure one match register for each match event by setting REGMODE_L bits 0, 1, 2, 3, 4 to 0. This is the default.
Define match values	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.

Table 183. SCT configuration example...continued

Configuration	Registers	Setting
Configure states in which event 2 is enabled	EV2_STATE	Set STATEMSK2 bit 0 to 1. Set all other bits to 0. Event 2 is enabled in state 0.
Configure states in which event 3 is enabled	EV3_STATE	Set STATEMSK3 bit 1 to 1. Set all other bits to 0. Event 3 is enabled in state 1.
Configure states in which event 4 is enabled	EV4_STATE	Set STATEMSK4 bit 1 to 1. Set all other bits to 0. Event 4 is enabled in state 1.
Configure states in which event 5 is enabled	EV5_STATE	Set STATEMSK5 bit 1 to 1. Set all other bits to 0. Event 5 is enabled in state 1.

24 Standard counter/timers (CTIMER0 - 4)

The five standard timers, CTIMER0, CTIMER1, CTIMER2, CTIMER3, and CTIMER4 are available on all MCX W23 devices.

24.1 Features

- Each is a 32-bit counter/timer with a programmable 32-bit prescaler. 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 the section "CTimer 0 - 4 clock source select (CTIMERCLKSELa)" in the Register description ([Ref. 5](#)).
- When an input signal transitions, up to four 32-bit captures can take a snapshot of the timer value. Optionally, a capture event may also generate an interrupt. The number of capture inputs for each timer that is available on the device pins may vary by device.
- The timer and prescaler may be configured to be cleared on a designated capture event. This feature permits easy pulse-width measurement by clearing the timer on the leading edge of an input pulse and capturing the timer value on the trailing edge.
- Four 32-bit match registers that allow:
 - Continuous operation with optional interrupt generation on match
 - When counter is reset, optional auto-reload from match shadow registers
 - 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 is available on the 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 is available on the device pins may vary by device).
- Up to two match registers can be used to generate DMA requests. These registers are connected to DMA trigger inputs on this device.

24.2 Basic configuration

- Select clock source (see the section "CTimer 0 - 4 clock source select (CTIMERCLKSELa)" in the Register description ([Ref. 5](#)).
- To enable clocks to timers that are used, set the appropriate bits using the AHBCLKCTRL registers (see the sections "AHB clock control 1 (AHBCLKCTRL1)" and "AHB clock control 2 (AHBCLKCTRL2)" in the Register description ([Ref. 5](#)).
- Clear the timer reset using the PRESETCTRL registers (see the sections "Peripheral reset control 1 (PRESETCTRL1)" and "Peripheral reset control 2 (PRESETCTRL2)" in the Register description ([Ref. 5](#)). Bit positions in the reset control registers match the bit positions in the clock control registers.
- Pins: Select timer pins and pin modes as required using the relevant IOCON registers (see [Section 15](#)).
- Interrupts: For match and capture events, see registers MCR (section "Match control register (MCR)" and CCR (section "Capture control register (CCR)" in the Register description ([Ref. 5](#)). Interrupts are enabled in the NVIC using the appropriate interrupt set enable register. For interrupt connections, see [Section 3.2.1](#).
- DMA: Some timer match conditions can be used to generate timed DMA requests (see [Section 22](#)).

24.3 General description

Each counter/timer is designed to count cycles of the APB bus clock or an externally supplied clock. Optionally, they can 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. When the counter is reset to zero, all match registers can be auto-reloaded from a companion shadow register. This option allows the modification of the match values for the next counter cycle without the risk of disrupting the PWM waveforms during the current cycle. When enabled, match reload occurs when the counter is reset due to a match event or to a write to bit 1 of the timer control register (TCR).

24.3.1 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. One of the pins with a capture function generates the capture signal. Each capture signal is connected to one capture channel of the timer.

The Counter/Timer block can select a capture signal as a clock source instead of the APB bus clock. For more details, see the section "Count control register (CTCR)" in the Register description ([Ref. 5](#)).

24.3.2 Match outputs

When a match register equals the timer counter (TC), the corresponding match output can 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.

24.3.3 Applications

- Interval timer for counting internal events
- Pulse width modulator via match outputs
- Pulse width demodulator via capture input
- Free running timer

24.3.4 Architecture

[Figure 72](#) shows the block diagram for the timers.

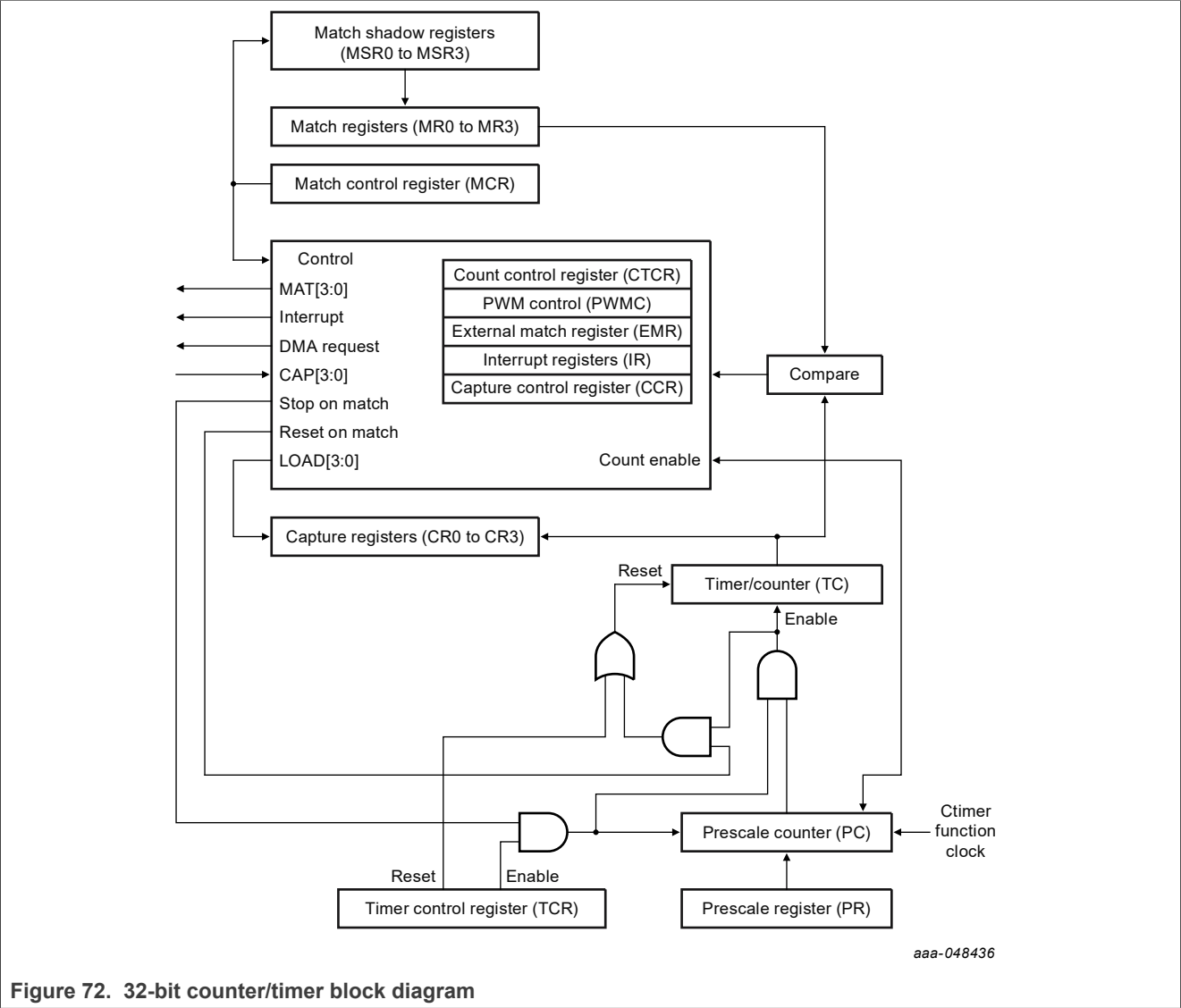


Figure 72. 32-bit counter/timer block diagram

24.3.5 Peripheral input multiplexers for CTimers

See [Section 18](#).

[Figure 46](#) shows the pin assignments for the peripheral input multiplexer.

24.4 Pin description

[Table 184](#) gives a brief summary of each of the Timer/Counter related pins.

Table 184. Timer/counter pin description

Pin	Type	Description
CTIMER0_INP22:0	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. The timer/counter block can select a capture signal as a clock source instead of the APB bus clock. For more details, see the section "Count control register (CTCR)" in the Register description (Ref. 5).
CTIMER0_MAT3:0 CTIMER1_MAT3:0 CTIMER2_MAT3:0 CTIMER3_MAT3:0 CTIMER4_MAT3:0	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.

24.4.1 Multiple CAP and MAT pins

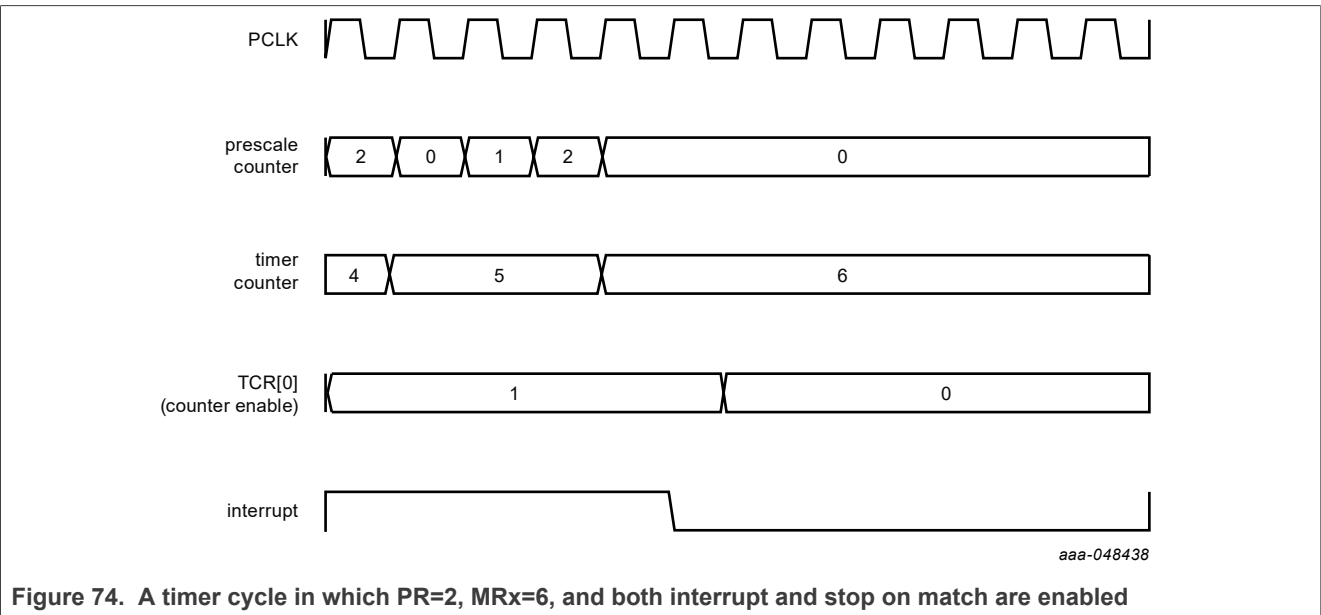
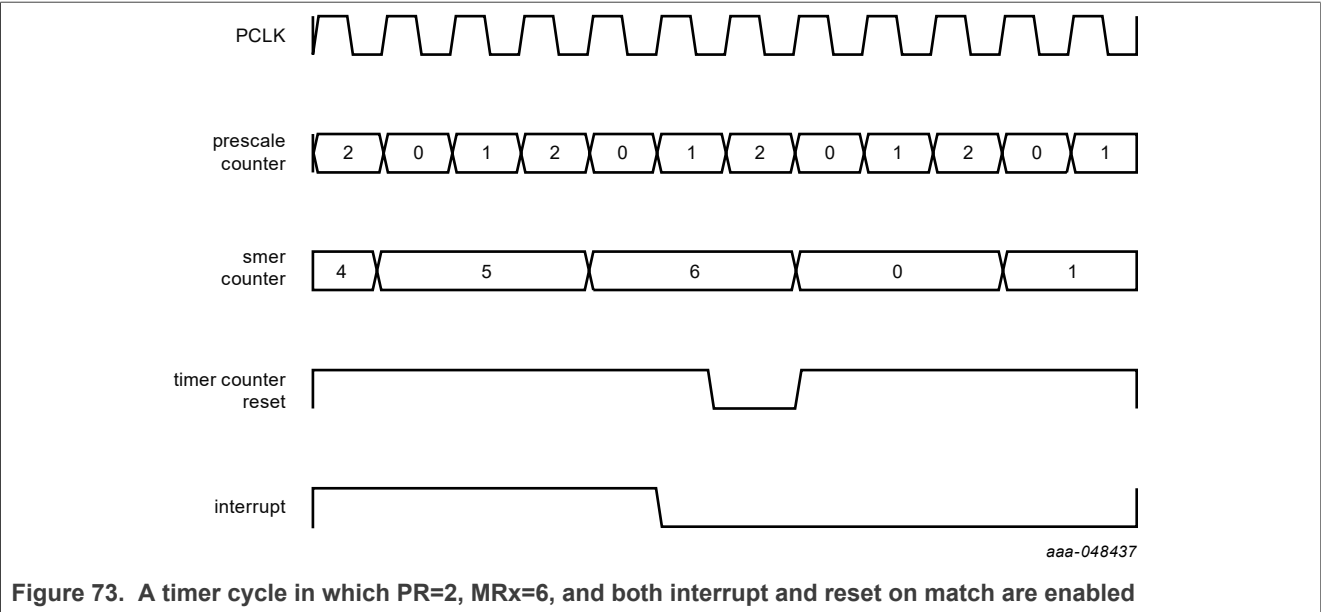
Software can select from multiple pins for the CAP or MAT functions in the IOCON registers. [Section 15](#) describes the IOCON registers.

Note: Match conditions may be used internally without using a device pin.

24.5 Functional description

[Figure 73](#) shows a timer configured to reset the count and generate an interrupt on match. The prescaler 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. It 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 74](#) 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.



24.5.1 Rules for single edge controlled PWM outputs

- 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.
- Each PWM output goes HIGH when its match value is reached. If no match occurs (that is, the match value is higher than the PWM cycle length), the PWM output remains continuously LOW.
- If a match value higher than the PWM cycle length is written to the match register and the PWM signal is HIGH already, the PWM signal is cleared with the start of the next PWM cycle.
- If a match register contains the same value as the timer reset value (the PWM cycle length), the PWM output is reset to LOW on the next clock tick after the timer reaches the match value. So, the PWM output always

consists of a one-clock-tick-wide positive pulse with a period that the PWM cycle length (that is, the timer reload value) determines.

If a match register is set to zero, the PWM output goes to HIGH the first time that the timer goes back to zero. It stays 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. The exception is 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.

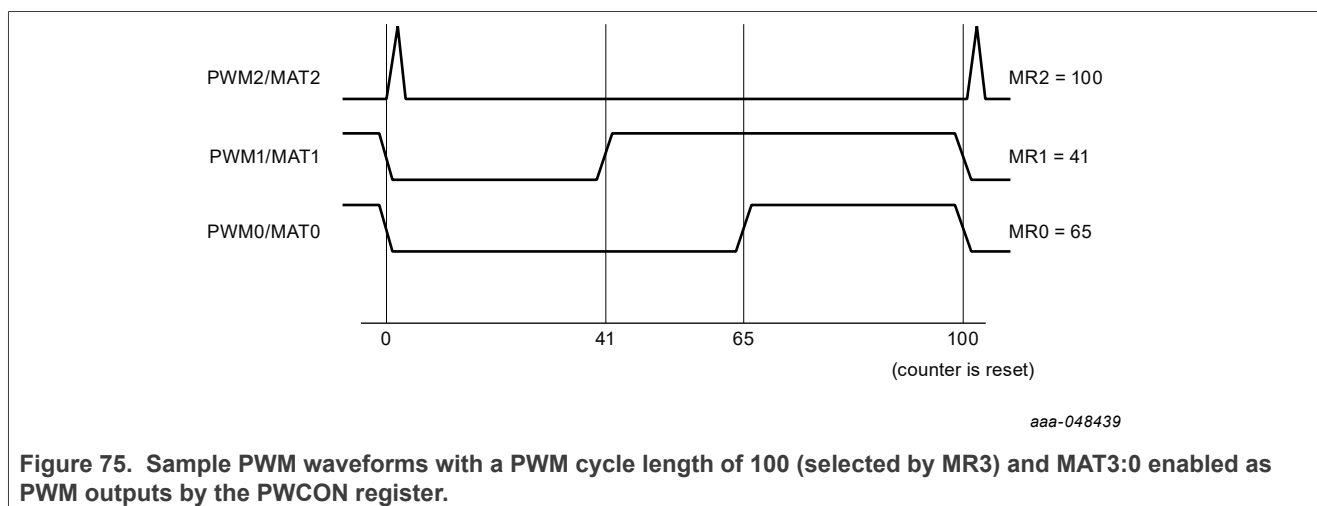


Figure 75. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

24.5.2 DMA operation (DMA0 and DMA1)

A match of the timer counter (TC) register value to either match register 0 (MR0) or match register 1 (MR1) generates DMA requests. It is not connected to the operation of the match outputs that the EMR register controls. Each match sets a DMA trigger flag, which is connected to the DMA controller. 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 the section Interrupt register (IR)" in the Register description ([Ref. 5](#))). When the DMA controller handles a DMA request, it is cleared automatically.

Note: Because timer DMA requests are generated when the timer value equals the related match register value, DMA requests are always generated when the timer is running. The exception is when 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.

25 Micro-tick timer (UTICK)

The Micro-tick timer is available on all MCX W23 devices.

25.1 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 when to start.
- Interrupt or software polling.
- Four capture registers that external pin transitions can trigger.

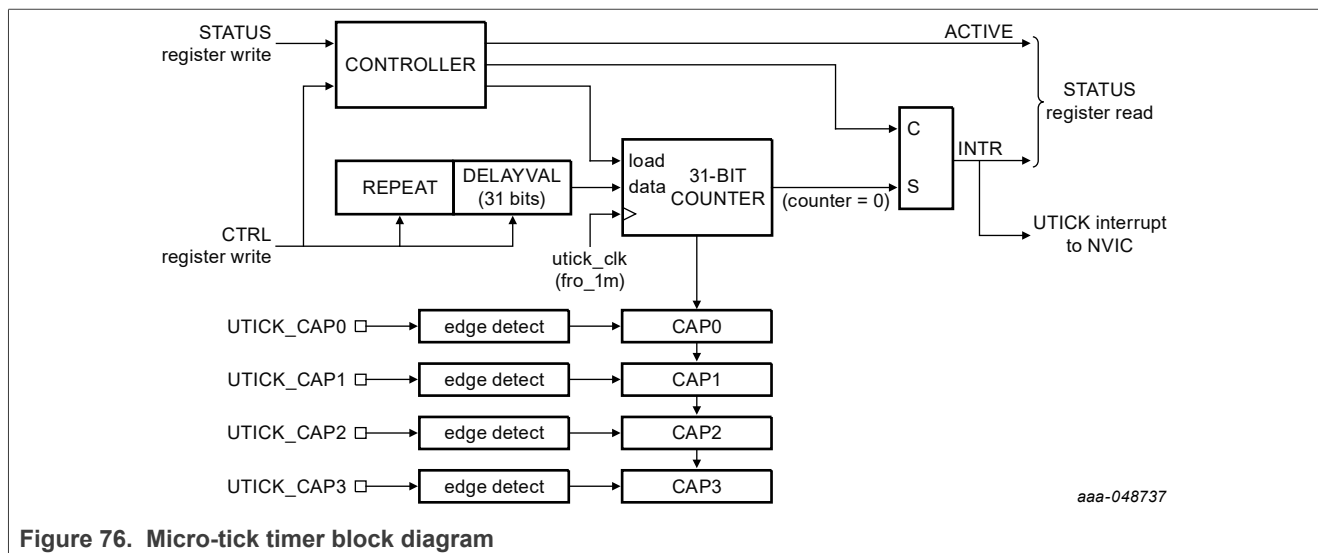
25.2 Basic configuration

Configure the micro-tick timer as follows:

- To enable the clock to the micro-tick timer register interface, set the UTICK bit in the AHBCLKCTRL1 register.
- The micro-tick timer provides an interrupt to the NVIC (see [Section 3](#)).
- To enable micro-tick timer interrupts for waking up from deep-sleep, use the low-power API provided `Power_EnterDeepSleep` (see [Section 14](#)).
- Configure the pin functions of any micro-tick timer capture pins that are used via IOCON (see [Section 15.3.1](#)).
- Enable the 1 MHz clock by programming bit 4 in the Syscon `CLOCK_CTRL` register (see the section "Clock control register (`CLOCK_CTRL`)" in the Register description ([Ref. 5](#)).

25.3 General description

[Figure 76](#) shows a conceptual view of the micro-tick timer.



25.4 Pin description

[Table 185](#) gives a summary of pins related to the micro-tick timer.

Table 185. Micro-tick timer pin description

Pin	Port Pins	Type	Description
UTICK_CAP0	PIO0_5 or PIO0_11	Input	Capture inputs. The selected transition on a capture pin can be configured to load the related CAP register with the value of counter.
UTICK_CAP1	PIO0_6 or PIO0_12		
UTICK_CAP2	PIO0_7 or PIO0_13		
UTICK_CAP3	PIO0_8 or PIO1_14		

26 Multirate timer (MRT)

The MRT is available on all MCX W23 devices.

26.1 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

26.2 Basic configuration

Configuration of the MRT is accomplished as follows:

- In the AHBCLKCTRL1 register (see the section "AHB clock control 1 (AHBCLKCTRL1)" in the Register description ([Ref. 5](#)), set the MRT bit to enable the clock to the register interface.
- Clear the MRT reset using the PRESETCTRL1 register (see the section "Peripheral reset control 1 (PRESETCTRL1)" in the Register description ([Ref. 5](#)).
- The global MRT interrupt is connected to an interrupt slot in the NVIC (see the section "Main clock B source select (MAINCLKSELB)" in the Register description ([Ref. 5](#)).
- It is recommended that the MRT counters are stopped before entering deep-sleep or 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.

26.3 Pin description

The MRT is not associated with any device pins.

26.4 General description

The multirate 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 26.4.1](#))
- One-shot interrupt mode (see [Section 26.4.2](#))
- One-shot stall mode (see [Section 26.4.3](#))

The modes for each timer are set in the control register of the timer (see the section "MRT control register (CTRL0 - CTRL3)" in the Register description ([Ref. 5](#)).

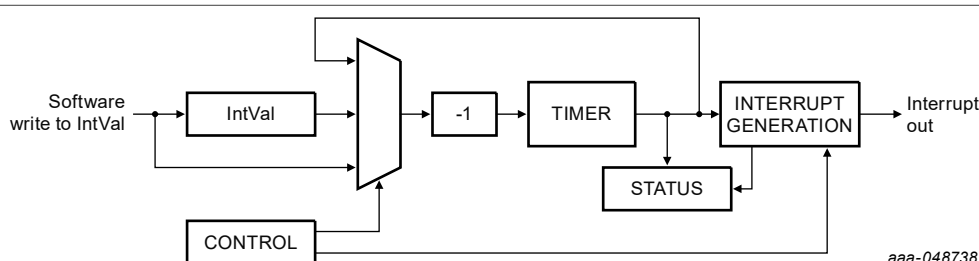


Figure 77. MRT block diagram

26.4.1 Repeat interrupt mode

The repeat interrupt mode generates repeated interrupts after a selected time interval. This mode can be used for software-based PWM or PPM applications.

When the timer n is in idle state, writing a non-zero value IVALUE to the INTVAL n register immediately loads the time interval value IVALUE – 1. The timer begins to count down from this value. When the timer reaches zero, an interrupt is generated. The value in the INTVAL n register IVALUE – 1 is reloaded automatically and the timer starts to count down again.

While the timer runs 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 INTVAL n register and setting the LOAD bit to 0. When the timer reaches zero, an interrupt is generated. 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 INTVAL n register and setting the LOAD bit to 1. The timer immediately starts to count down from the new timer interval value. When the timer reaches zero, an interrupt is generated.
- Stop the timer at the end of time interval by writing a 0 to the INTVAL n register and setting the LOAD bit to 0. When the timer reaches zero, an interrupt is generated.
- Stop the timer immediately by writing a 0 to the INTVAL n register and setting the LOAD bit to 1. When the INTVAL n register is written, no interrupt is generated.

26.4.2 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 INTVAL n register immediately loads the time interval value IVALUE – 1. The timer starts to count down. When the timer reaches zero, an interrupt is generated. The timer stops and enters the idle state.

While the timer runs in the one-shot interrupt mode, the following actions can be performed:

- Update the INTVAL n 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. When the TIME_INTVAL n register is updated, no interrupt is generated.
- Write a 0 to the INTVAL n register and set the LOAD bit to 1. The timer immediately stops counting and moves to the idle state. When the INTVAL n register is updated, no interrupt is generated.

26.4.3 One-shot stall mode

The one-shot stall mode is similar to the one-shot interrupt mode, except that it is intended for very short delays. For example, when the delay required 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). While the bus write operation is stalled, the MRT times the requested delay. When the delay is complete, the write is concluded. No interrupt or status polling is necessary.

When a short delay between two software controlled events is required or when a delay is expected before software can continue, the bus-stall mode can be used. Since, in this mode, there are no bus transactions while the MRT is counting down, the CPU consumes minimum power during that time.

Note: *The bus stall mode provides a minimum time period between the execution of the instruction that performs the write to INTVAL and the continuation of the software. Other system events, such as interrupts or other bus masters accessing the APB bus where the MRT resides, can cause the delay to be longer.*

27 Real-time clock (RTC)

The RTC is available on all MCX W23 devices.

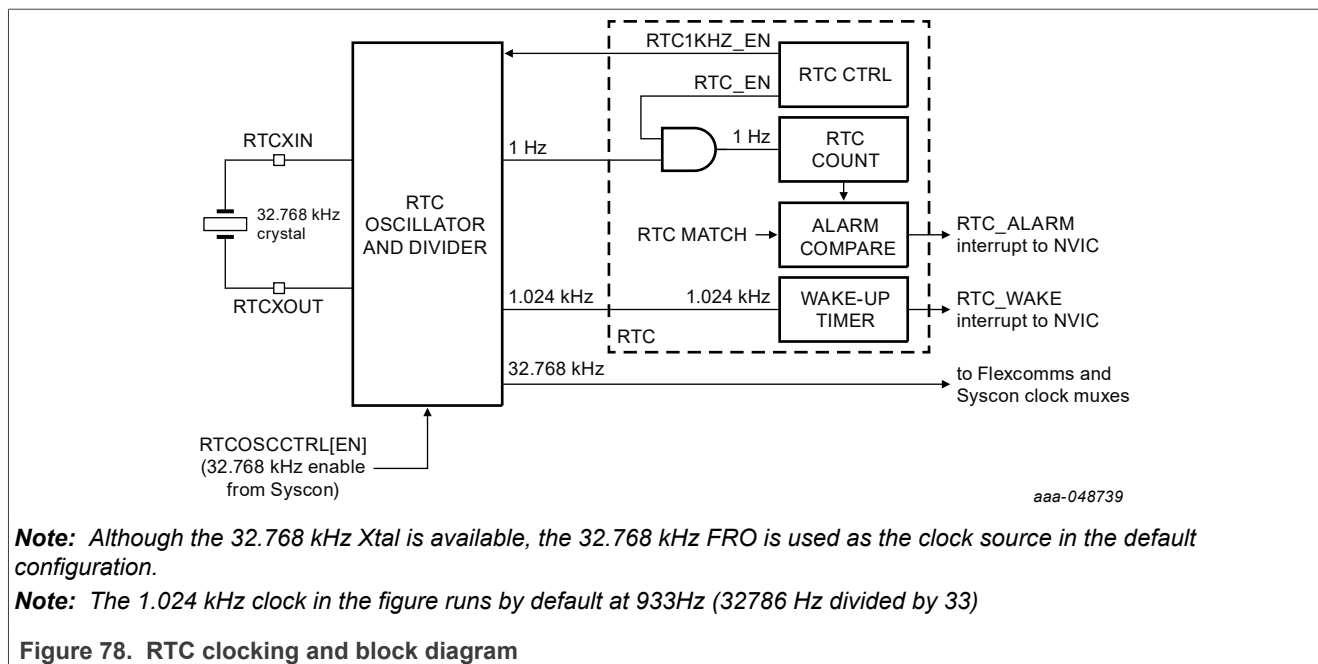
27.1 Features

- The RTC oscillator has the following clock outputs:
 - 32.768 kHz clock, selectable for system clock and CLKOUT pin.
The 32.768 kHz clock can be the 32.768 kHz Xtal or the 32.768 kHz free-running oscillator (FRO).
 - 1 Hz clock for RTC timing.
 - 993 Hz clock for high-resolution RTC timing.
- 32-bit, 1 Hz RTC counter and associated match register for alarm generation.
- 15-bit, 32.768 kHz subsecond counter.
- Separate 16-bit high-resolution/wake-up timer clocked at 993 Hz for 1 ms resolution with a more than 1 minute maximum timeout period.
- RTC alarm and high-resolution/wake-up timer timeouts, each generating independent interrupt requests that go to one NVIC channel. These timeouts 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 if a power failure occurs, provided there is battery backup.

27.2 Basic configuration

Configure the RTC as follows:

- To enable the clock to the RTC register interface and peripheral clock, use the AHBCLKCTRL0 register (see the section "AHB clock control 0 (AHBCLKCTRL0)" in the Register description ([Ref. 5](#)).
- For RTC software reset, use the RTC CTRL register (see the section "RTC control register (CTRL)" in the Register description ([Ref. 5](#)). The RTC is only reset at the initial power-up of the device or when an RTC software reset is applied. Other system resets do not initialize it.
- The RTC provides an interrupt to the NVIC for the RTC_WAKE and RTC_ALARM functions (see [Section 3](#)).
- To enable the RTC interrupts for waking up from deep-sleep and power-down modes, enable the interrupts using low-power APIs. Also, enable the interrupts 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 the section "RTC control register (CTRL)" in the Register description ([Ref. 5](#)).
- If enabled, the RTC and its oscillator continue to run in all reduced power modes as long as power is supplied to the device. So, the 32.768 kHz output is always available to be enabled for syscon clock generation (see [Table 10](#)). When enabled, the 32.768 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). When the 1 Hz output is enabled, the 1.024 kHz output for the high-resolution wake-up timer can be enabled in the RTC CTRL register (RTC1KHZ_EN bit).
- If another part of the system uses the 32.768 kHz output of the RTC, enable it via the EN bit in the RTCOSCCTRL register.



27.2.1 RTC timers

The RTC contains two counters:

- The main RTC timer
This 32-bit timer uses a 1 Hz clock. It is intended to run continuously as a real-time clock. When the timer value reaches a match value, an interrupt is raised. If enabled, the alarm interrupt can also wake up the part from any low-power mode.
- The high-resolution/wake-up timer
This 16-bit timer uses a 993 Hz clock. It operates as a one-shot down timer. When the timer is loaded, it starts counting down to 0 at which point an interrupt is raised. If enabled, the interrupt can wake up the part from any low-power mode. This timer is used for timed wake-up from deep-sleep or deep power-down modes. To conserve power, the high-resolution wake-up timer can be disabled, if not used.

27.3 General description

27.3.1 Real-time clock

The real-time clock is a 32-bit up-counter which software can clear or initialize. When 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, when the counter value equals the value programmed into the associated 32-bit match register, generate an alarm interrupt to the processor.

If the part is in one of the reduced-power modes (deep-sleep mode, power-down mode, or deep power-down mode), an RTC alarm interrupt can also wake up the part to exit the power mode and begin normal operation.

27.3.2 RTC subsecond counter

Table 186. RTC subsecond counter register (SUBSEC, offset 0x10)

Bit	Symbol	Description	Reset value
14:0	SUBSEC	A read reflects the current value of the 32.768 kHz subsecond counter. When the SUBSEC_ENA bit in the RTC_CONTROL register is low, this counter is cleared. Upcounting at a 32.768 kHz rate commences at the start of the next one-second interval after the SUBSEC_ENA bit is set. After exiting deep power-down mode or after the main RTC module is disabled and reenabled, this counter must be reenabled. On modules not equipped with a subsecond counter, this register reads back as all zeroes.	0
31:15	-	reserved. Read value is undefined, only zero must be written.	-

27.3.3 High-resolution/wake-up timer

The time interval required for many applications, including waking up the part from a low-power mode, often demands a greater degree of resolution than the one-second minimum interval that the main RTC counter affords. For these applications, a higher-frequency secondary timer has been provided.

This secondary timer is an independent, standalone wake-up or general-purpose timer for timing intervals of up to 64 seconds with approximately one millisecond of resolution.

If it is enabled, the high-resolution/wake-up timer is a 16-bit down counter, which is clocked at a 993 Hz rate. Writing any nonzero value to this timer automatically enables the counter and launches a countdown sequence. When the counter is used as a wake-up timer, this write can occur just before entering a reduced power mode.

When a starting count value is loaded, the high-resolution/wake-up timer turns on, counts from the preloaded value down to zero, generates an interrupt and/or a wake-up command, and then turn itself off until a subsequent software write relaunched it.

27.3.4 General purpose backup registers

The general purpose registers retain data through the deep power-down mode. Only a complete removal of power from the chip, or entering power-off mode, or a software reset of the RTC will clear the general purpose registers.

The RTC module offers this set of registers as backup storage. These registers may be used to store critical data through deep power-down mode.

27.3.5 RTC power

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 subsecond timer can continue running in deep power-down mode.

27.4 Pin description

[Table 187](#) gives a summary of pins related to the RTC.

Table 187. RTC pin description

Pin	Type	Description
XI32K	input	32 kHz crystal oscillator input
XO32K	output	32 kHz crystal oscillator output.

28 System tick timer (SYSTICK)

There are two system tick timers (SysTick timer). One is secured and the other is non-secured.

Each tick timer has its own calibration that the system configuration provides.

28.1 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 the system configuration

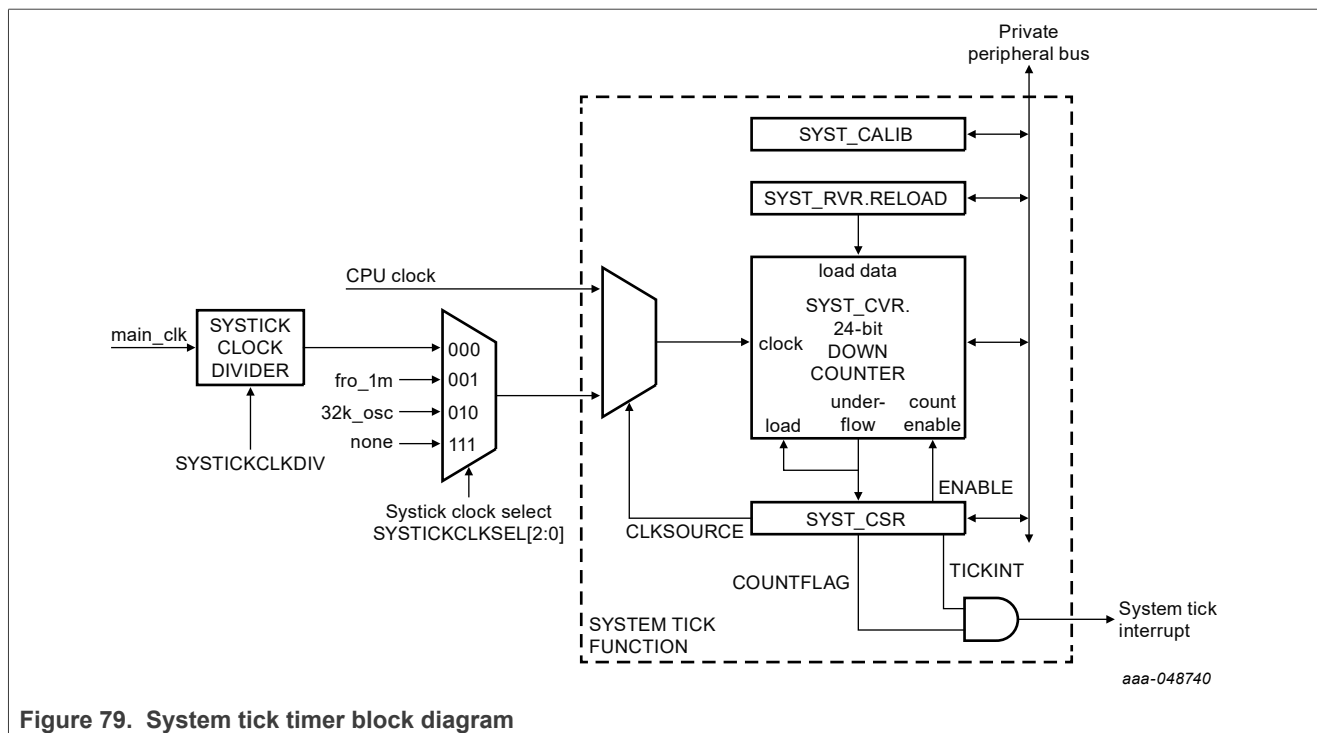
28.2 Basic configuration

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).

28.3 General description

See [Figure 79](#) for the block diagram of the SysTick timer.



The SysTick timer is a part of the Cortex-M33. It is intended to generate a fixed 10 ms interrupt that an operating system or other system management software can use.

Since the SysTick timer is a part of the CPU0, 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 is completed within a set duration, as part of a dynamic clock management control loop.

For more information, see the Cortex-M33 DGUG ARM Cortex-M33 Devices Generic User Guide ([Ref. 1](#)).

28.4 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](#)). 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. So, its frequency depends on Main_Clk frequency and the clock divider settings driven from syscon register SYSTICKCLKDIV0 (CPU0). The Sys Tick register, SYST_CALIB, is a read-only register. To indicate the number of ticks needed for a 10 ms period, the field TENMS can be used. 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 registers, using the SKEW and NOREF fields.

28.5 Example timer calculations

To use the system tick timer, do the following:

1. To obtain the desired time interval, program the SYST_RVR register with the reload value calculated as shown below.
2. Clear the SYST_CVR register by writing to it. It ensures that, when the timer is enabled, the timer counts from the SYST_RVR value rather than an arbitrary value. The following examples illustrate the selection of SYSTICK timer reload values for different system configurations. All examples calculate an interrupt interval of 10 ms, as the SYSTICK timer is intended to be used. There are no rounding errors.

Option 1: external reference clock as system tick timer clock

- System tick timer clock = 8 MHz
- CPU clock = 32 MHz

Program the SYST_CSR register with the value 0x3, which selects the external clock source (see the sections "System tick timer divider for CPU0 (SYSTICKCLKDIV0)" and "System tick timer source select (SYSTICKCLKSEL0)" in the Register description ([Ref. 5](#)).

Use DIV = 0x3 of the SYSTICKCLKDIV0 (CPU0).

$\text{SYST\_RVR} = (\text{system tick timer clock frequency} \times 10 \text{ ms}) - 1 = (8 \text{ MHz} \times 10 \text{ ms}) - 1 = 80000 - 1 = 79999 = 0x0001\ 387F$

Option 2: CPU clock as system tick timer clock

- 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.

$\text{SYST\_RVR} = (\text{CPU clock frequency} \times 10 \text{ ms}) - 1 = (12 \text{ MHz} \times 10 \text{ ms}) - 1 = 120000 - 1 = 119999 = 0x0001 \text{ D4BF}$

29 Windowed watchdog timer (WWDT)

The watchdog timer is available on all MCX W23 devices.

29.1 Features

- Internally resets chip if not reloaded during the programmable timeout period.
- Optional windowed operation requires reload to occur between a minimum and maximum timeout period, both programmable.
- Optional warning interrupt can be generated at a programmable time before the watchdog timeout.
- Programmable 24-bit timer with internal fixed prescaler.
- 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. When enabled, requires a hardware reset or a watchdog reset to be disabled.
- If enabled, an incorrect feed sequence causes an immediate watchdog event.
- The watchdog reload/watchdog feed sequence can 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 always derived from the FRO_1MHz clock (see [Figure 80](#)) using the WDT clock divider register value in the SYSCON module (see the section "WDT clock divider (WDTCLKDIV)" in the Register description ([Ref. 5](#)) can divide. The accuracy of this clock is limited to $\pm 25\%$ over temperature, voltage, and silicon processing variations. To determine the actual watchdog frequency, use the frequency measure block (see [Section 11](#)).
- The watchdog timer can be configured to run in deep-sleep mode.
- Debug mode.

29.2 Basic configuration

Configuration of the WWDT is accomplished as follows:

- Configure WDTCLKDIV register (see the section "WDT clock divider (WDTCLKDIV)" in the Register description ([Ref. 5](#)). 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 the section "AHB clock control 0 (AHBCLKCTRL0)" in the Register description ([Ref. 5](#)).
- For waking up from a WWDT interrupt, enable the watchdog interrupt for wake-up using a low-power API.

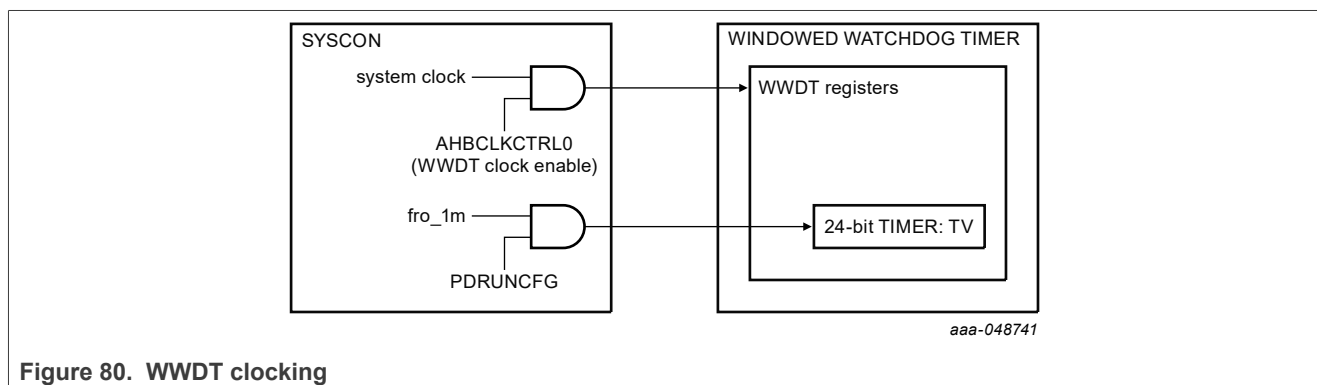


Figure 80. WWDT clocking

29.3 Pin description

The WWDT has no external pins.

29.4 General description

The purpose of the watchdog timer is to reset or interrupt the microcontroller within a programmable time if it enters an erroneous state. When enabled, a watchdog reset is generated if the user program fails to feed (reload) the watchdog within a predetermined time period.

When a watchdog window is programmed, an early watchdog feed is also treated as a watchdog event. It allows the prevention of situations where a system failure may still feed the watchdog. For example, application code can be stuck in an interrupt service that contains a watchdog feed. Setting the window such that it results in an early feed, generates a watchdog event, allowing a system recovery.

The watchdog consists of a fixed (divide by 4) prescaler and a 24-bit counter which, when clocked, decrements. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. So, 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 must be used in the following manner:

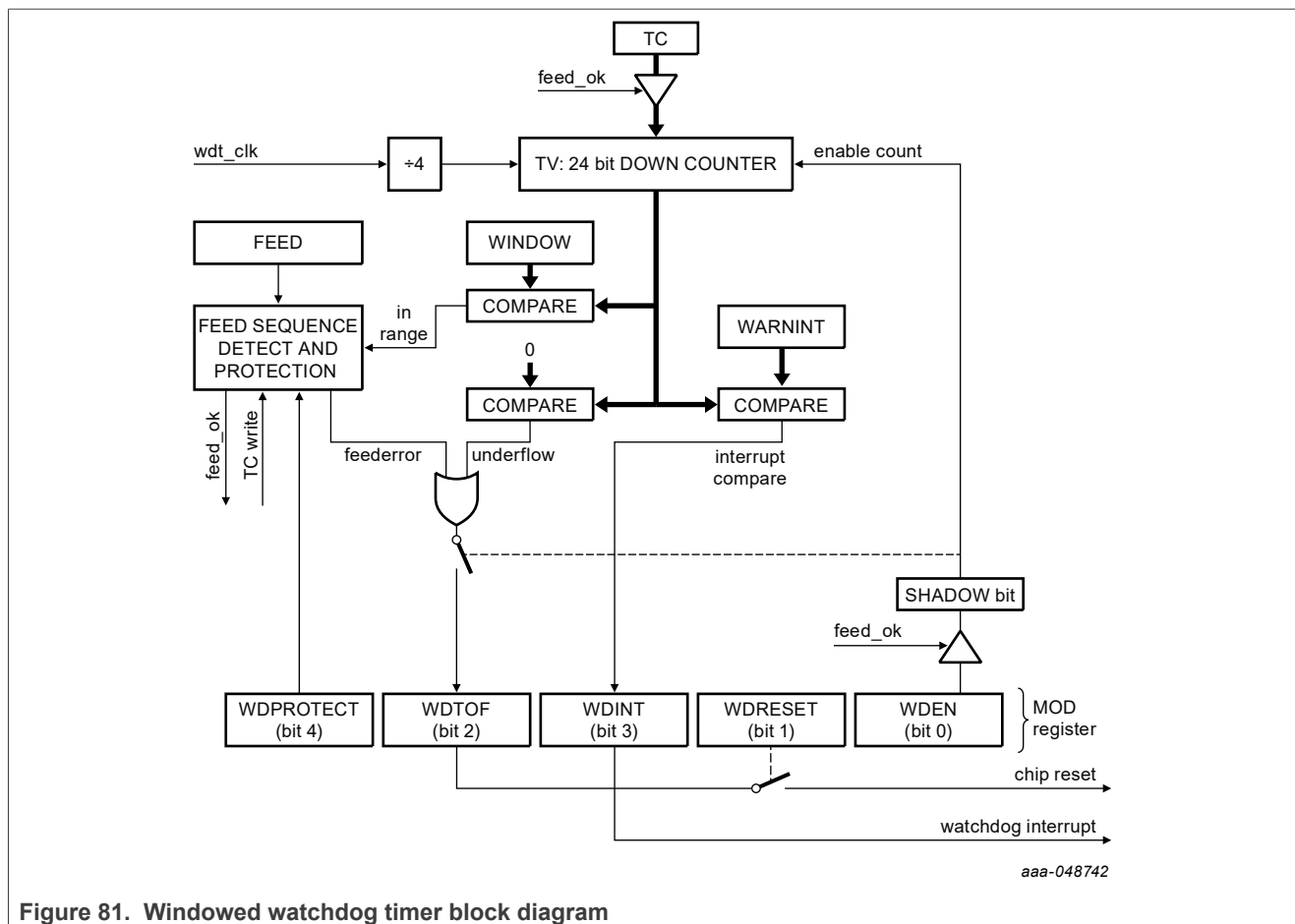
- Enable and configure the watchdog clock as described in [Section 29.2](#).
- Set the watchdog timer constant reload value in the TC register.
- Set the watchdog timer operating mode in the MOD register.
- Set a value for the watchdog window time in the WINDOW register if windowed operation is desired.
- 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.
- To prevent a watchdog event, the watchdog must be fed again before the watchdog counter reaches zero. 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 causes a reset and the counter reaches zero, the CPUs are reset, loading the stack pointer and program counter from the vector table as for an external reset. To determine if the watchdog has caused the reset condition, the watchdog timeout flag (WDTOF) can be examined. Software must clear the WDTOF flag.

When the watchdog timer is configured to generate a warning interrupt, the interrupt occurs when the counter is no longer greater than the value that the WARNINT register defines.

29.4.1 Block diagram

[Figure 81](#) shows the block diagram of the watchdog. The synchronization logic (APB bus clock to WDCLK) is not shown in the block diagram.



29.4.2 Clocking and power control

The watchdog timer block uses two clocks, the APB bus clock and the WDCLK. The APB bus clock is used for the APB accesses to the watchdog registers. It is derived from the system clock (see Figure 81). The WDCLK is used for the watchdog timer counting. It is derived from the 1 MHz FRO clock that can be divided.

The synchronization logic between the two clock domains is that when APB operations update the MOD and TC registers, the new value takes effect within 3 WDCLK cycles on the logic in the WDCLK clock domain.

When the watchdog timer counts on WDCLK, the synchronization logic first locks the value of the counter on WDCLK and then synchronizes it with the APB bus clock, so that the CPU can read the TV register.

Note: Because of the synchronization step, software must add a delay of three WDCLK clock cycles between the feed sequence and the time that the WDPROTECT bit is enabled in the MOD register. The length of the delay depends on the selected watchdog clock WDCLK.

29.4.3 Using the WWDT lock feature

To ensure that the WWDT always runs, the WWDT supports a lock feature which can be enabled:

- Performing the WWDT reload/WWDT feed sequence.

29.4.3.1 Changing the WWDT reload value

If WDPROTECT is set to 0b, a watchdog feed error is signaled when the FEED sequence is written (using void WWDT_Refresh()) while the watchdog timer value (TV) is greater than the programmed watchdog window value (WINDOW - using void WWDT_SetWindowValue()).

If WDPROTECT is set to 1b, a watchdog feed error is signaled when the FEED sequence is written (using void WWDT_Refresh()) while the watchdog window value (TV) is greater than the programmed watchdog window value (WINDOW) or greater than the programmed interrupt value (WARNINT - using void WWDT_SetWarningValue()).

A correct FEED sequence will reload the watchdog counter with the programmed TC value (using void WWDT_SetTimeoutValue()) and continue to count down from the reloaded value. To know the current value of the counter, you can read the timer value (TV) register.

Only a reset of any type can disable the reload overwrite lock mechanism (clearing WDPROTECT bit).

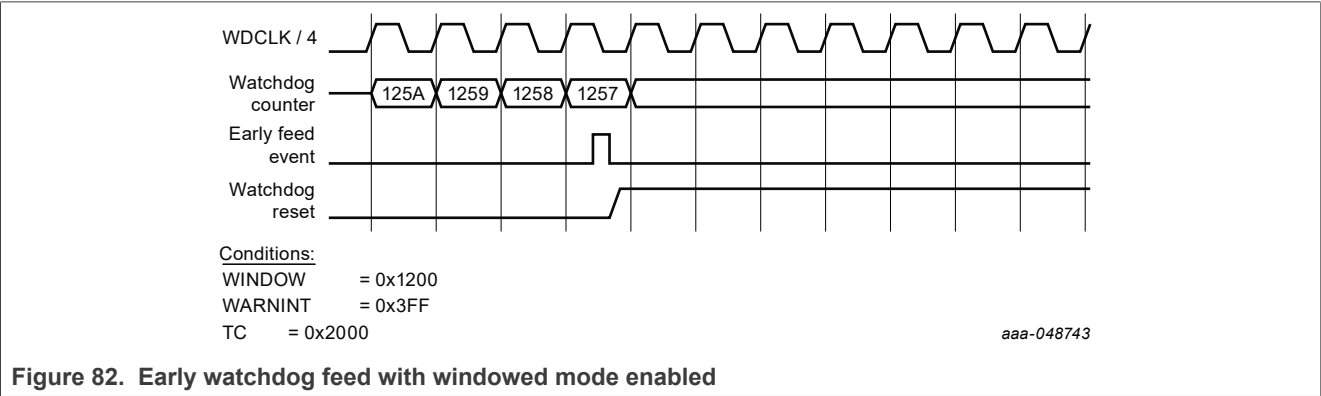
29.5 Functional description

The figures below illustrate several aspects of watchdog timer operation.

When the WWDT is enabled, the watchdog counter starts counting down from the programmed timeout value (TC) to zero. When watchdog counter value (TV) reaches zero, a watchdog reset is generated.

When a watchdog FEED event occurs (using void WWDT_Refresh()) at the expected moment, the watchdog counter value (TV) is reloaded with the programmed watchdog timeout value (TC).

[Figure 82](#) shows that a watchdog reset event is generated when a FEED event (using void WWDT_Refresh()) is executed while the current watchdog counter value (TV) is greater than the watchdog window value (WINDOW)



[Figure 83](#) shows that no watchdog reset is generated when the FEED event is executed while the current watchdog counter value (TV) is smaller than or equal to the watchdog window value (WINDOW)

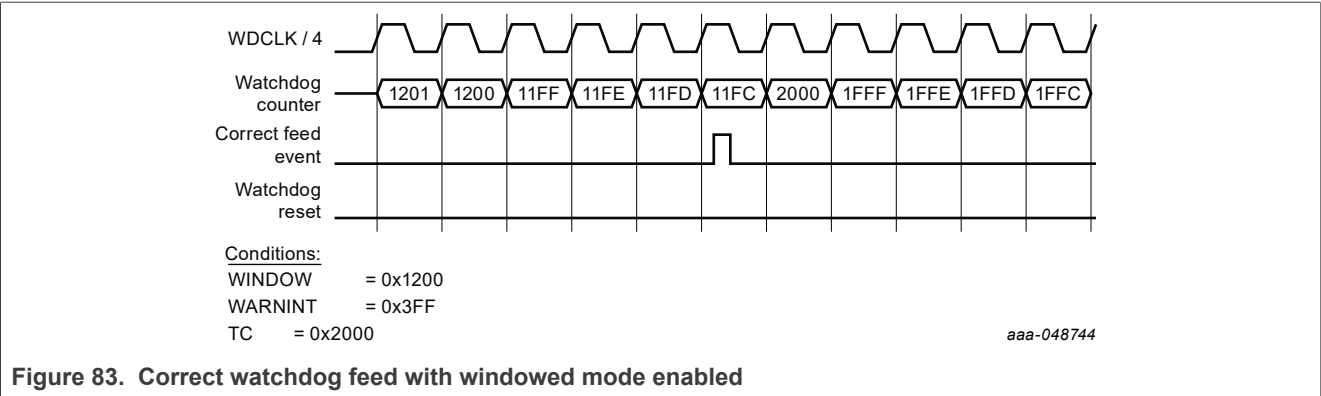


Figure 83. Correct watchdog feed with windowed mode enabled

Figure 84 shows that a watchdog warning interrupt is raised as long as the current watchdog counter value (TV) is smaller than the watchdog warning interrupt value (WARNINT).

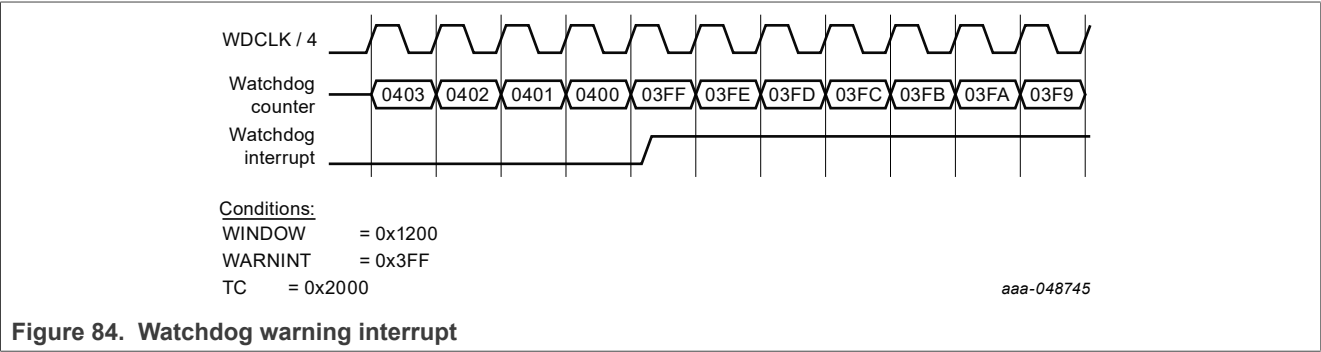


Figure 84. Watchdog warning interrupt

30 Code watchdog timer (CDOG)

The code watchdog timer is available on all MCX W23 devices.

30.1 Basic configuration

The CDOG provides two primary mechanisms for detecting code flow integrity as follows.

Secure counter (SECURE_COUNTER): Detects changed software in the execution flow.

- This counter (an accumulator) is loaded with an initial value. Then the runtime software issues ADD and SUB commands to increment/decrement the counter.
- Periodically, a secure counter value check is initiated by passing the expected value to the CDOG using the STOP and RESTART commands.
- If a mismatch is detected between the secure counter and the value passed to it, a side channel attack or some other suspicious activity may have changed the execution flow.

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 Pin description

The CDOG has no external pins.

30.3 General description

30.3.1 Register overview

CONTROL group of registers

The CONTROL group of the registers performs 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.

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.

STATE registers

The CDOG has two legal states: IDLE and ACTIVE.

Internally, four bits encode the two states. However, only two of the possible sixteen combinations are allowed.

After any reset (including a reset that the CWD itself generates), the module is in the 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 changes the state back to IDLE.

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.

FAULT counters

Each FAULT type has an associated counter, which increments when the fault is detected. The counters retain their value through a system reset. POR clears them.

FAULT flags

Each fault type has an associated flag that is set when the fault is detected. The flags retain their value through a system reset. POR or software clears them.

30.3.2 Operation

30.3.2.1 During code development debug

Suggestions for facilitating the code development and debug cycle:

- Use interrupt-on-fault (instead of reset) to maintain control while dialing in the timing and the counting values to be used.
- Direct-write the bits in the flags register to trigger faults, with the same result as hardware-triggered faults.
- Use the `DEBUG_HALT_CTRL` field to pause the Instruction Timer during the pausing of a debug session.

30.3.2.2 Model use cases

There is a strict sequence with which the CDOG is configured, activated, and serviced:

1. Write to the RELOAD register with the value corresponding to the current code section.
2. To generate a reset and lock the lock field, write a control word to the CONTROL register, where each fault type is enabled (or not). Depending on the value that is written to the `IRQ_PAUSE` field, the instruction timer may be kept running continuously or only during non-IRQ execution (paused during interrupt processing).
3. Activate the module by writing to the START command register. The initial value for the `SECURE_COUNTER` is the value written. The instruction timer starts decrementing from the RELOAD value on every clock cycle immediately.
4. At strategically chosen way points in the code flow of the application, update the `SECURE_COUNTER` by issuing ADD or SUB commands.
5. When the way point, that corresponds to the number of executed instructions that the RELOAD value represents (the end of the current code section), is reached, write the expected value of the `SECURE_COUNTER` to the STOP command register. Assuming that the written value compares exactly to the contents of the `SECURE_COUNTER`, the instruction timer stops and the process can recommence for the next code section, by repeating steps 1 through 5.

Another example of a configuration and start procedure:

1. Write the instruction timer reload value to the RELOAD register. To provide a buffer for high values, write a very high number.
2. To generate a reset and then lock the lock field, write a control word to the CONTROL register, where each fault type is enabled (or not).
3. Activate the module by Writing to the START command register. The initial value for the `SECURE_COUNTER` is the value written. The instruction timer starts decrementing from the RELOAD value on every clock cycle immediately.
4. At strategically chosen way points in the code flow of the application, update the `SECURE_COUNTER` by issuing ADD or SUB commands.
5. Within the execution flow, the application must always be aware of the approach-to-zero of the instruction counter. To determine the expended time, the instruction timer may require regular reading. Before the `TIMER` reaches 0, software must service the CDOG by writing the `SECURE_COUNTER` expected value to the RESTART command register. Assuming that the value written compares exactly to the contents of

the SECURE ACCUMULATOR, the instruction timer is reloaded with the value in the RELOAD register as it decrements toward 0.

30.3.3 Faults

There are six types of faults that are detectable (TIMEOUT, MISCOMPARE, SEQUENCE, CONTROL, STATE, ADDRESS). To generate either a system reset, an interrupt, or nothing, each type can be controlled individually.

Note: *Illegal values in the various bitfields of the CONTROL register itself can only be enabled to generate either a system reset or nothing.*

Interrupts and resets can be enabled simultaneously, but each fault type can only contribute to one or the other, not to both. Interrupts are available as an alternative to a 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.

TIMEOUT

A TIMEOUT fault is raised when the instruction timer times out (that is, reaches '0').

MISCOMPARE

A MISCOMPARE fault is raised when either a STOP or a RESET command is issued and the value passed in the instruction does not match (that is, does not compare with) the content of the SECURE_COUNTER.

SEQUENCE

A SEQUENCE fault is raised when the prescribed sequence of interactions between SW and CDOG is violated.

CONTROL

A CONTROL fault is raised if any field of the CONTROL register contains an illegal value.

STATE

A STATE fault is raised if the internal state machine ends up in one of the fourteen illegal combinations that the four-bit coding of the STATE can encode.

ADDRESS

An ADDRESS fault is raised when the CPU accesses any address in the register space of the module, which is not legally defined.

30.3.4 Flags

The CDOG has one flag for each fault type, which is accessible through the FLAGS register. A flag is set when its associated fault is detected. Software can clear a flag. The flags themselves (when enabled) generate a system reset or interrupt outputs of the module.

The flags retain their value through a system reset (including a reset that CDOG generates). However, a POR resets them. 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 does not contribute to a system reset or an interrupt. It is set to '1' after POR. SW can clear the PORF flag with a write of '1' to its bit position. However, software cannot set it.

30.3.5 Counters

Each fault type has a counter that, when a fault of the specified type is detected, increments. POR, not a system reset, clears the counters. So, to reveal the behavior patterns of a specific type of attack, statistics can be built up over many code watchdog resets.

The CDOG supports the following faults:

- Before the START command is issued, RELOAD must be written. Failure to do so generates an illegal sequence fault.
- RELOAD can only be written in the IDLE state. Attempts to write to RELOAD in ACTIVE state generate an illegal sequence fault.
- START can only be written in the IDLE state and only if RELOAD has first been written. Attempts to write to START in ACTIVE state generate an illegal sequence fault.
- One clock before the instruction timer reaches 0, the TIMEOUT fault is generated.
- 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 the 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 the 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, SEQUENCE_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 change to the CONTROL[IRQ_PAUSE] bit or the IRQ_PAUSE 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 the CONTROL and the COMMAND register groups within the 4K byte address space.

31 OS event timer (OSTIMER)

The OS event timer is available on all MCX W23 devices.

31.1 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 that a CPU command triggers, 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 mode.

31.2 Basic configuration

Configure the OS event timer as follows:

- To enable the clock to the OS event timer register interface, use the AHBCLKCTRL1 register in SYSCON. To select and enable the OS event timer peripheral clock (1 MHz or 32.768 kHz), use the OSTIMER register in PMC. When the FRO/XTAL 32.768 kHz output clock is selected, it must also be enabled in the RTCOSC32K register in PMC (see the Register description ([Ref. 5](#))).
- To clear the reset to the OS event timer, use the PRESETCTRL1 register in SYSCON (see the Register description ([Ref. 5](#))). Clear the 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 the OS event timer interrupt in the NVIC.
- For the OS event timer software reset, use the OSTIMER register (see the Register description ([Ref. 5](#))).
- The OS event timer provides an interrupt to the NVIC.
- This module is placed in the always-on domain. So, it can run in all low-power modes, including deep power-down mode. 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.

31.3 Pin description

The OS event timer is not associated with any device pins.

31.4 General description

The OS event timer incorporates one central 42-bit timer ("EVTimer"), separate 42-bit match and capture registers, and a maskable IRQ/wake-up request. [Figure 85](#) shows a conceptual view of the OS event timer.

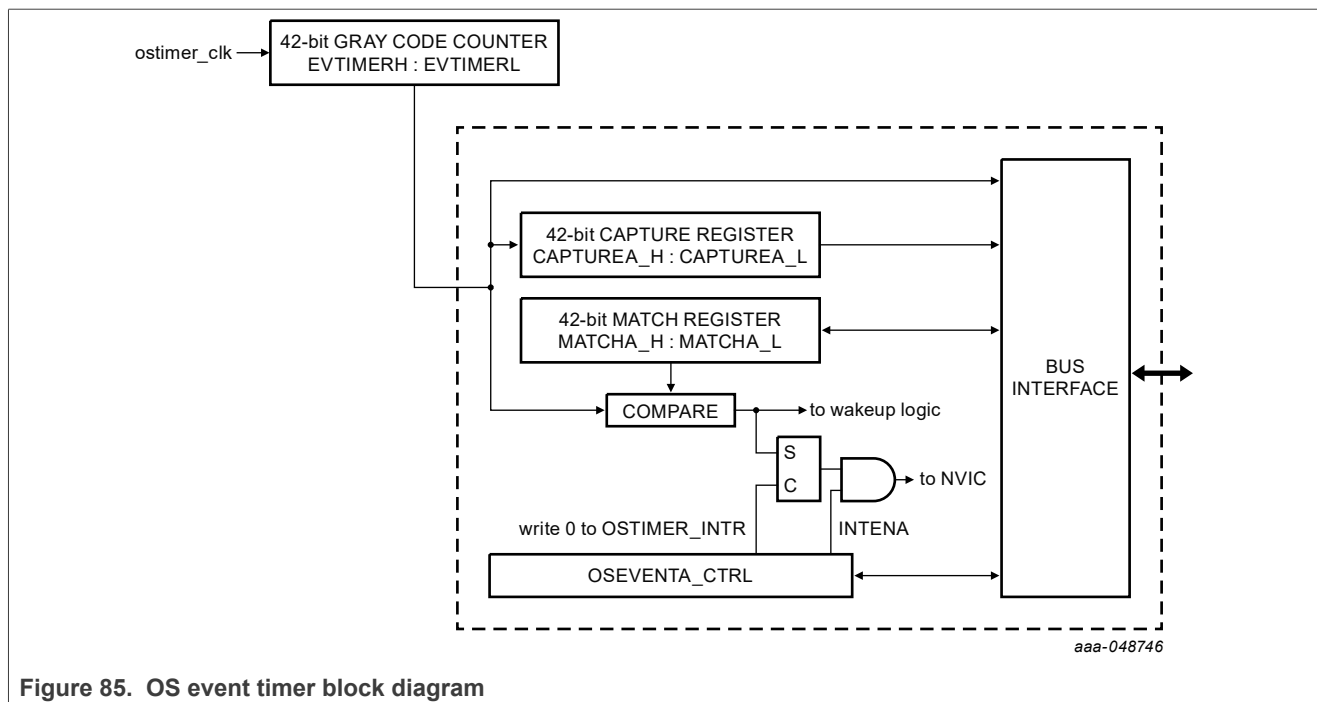


Figure 85. OS event timer block diagram

31.4.1 Central event/timestamp timer

The 42-bit central EVTimer is initialized on a full-system POR. It counts up continuously. The typical clock for this timer is a 32.768 kHz clock.

The central EVTimer is implemented as a gray-coded counter and can be read from capture registers.

31.4.2 Match, capture, and interrupt generation

The timer includes capture registers, match registers, and a control register.

31.4.2.1 Capture registers

42-bits of capture values are available in the Capture_L and Capture_H registers. A capture command that the CPU issues (an Arm "Set Event" intrinsic instruction or in CMSIS C-coding by the function "`__SEV()`;" issues the capture command) causes the current value of the main EVTimer to be stored in the capture registers. The same clock as the associated clocks the CPU. Use of gray encoding eliminates issues associated with the asynchronous transfer from the EVTimer to the capture registers.

31.4.2.2 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 sets 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 reflect the value in the bus clock-based shadow registers. So, a read immediately following a write always reflects the value just written.

32 Flexcomm interface serial communication (FLEXCOMM0 - 2)

Multiple Flexcomm interfaces are available on all MCX W23 devices.

32.1 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.

32.2 Features

Each Flexcomm interface provides a choice of peripheral functions, one of which the user must choose 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 two slave selects.
- I²C, including separate master, slave, and monitor functions.
- Data for USART and SPI traffic uses the Flexcomm Interface FIFO. The I²C function does not use the FIFO.

32.3 Basic configuration

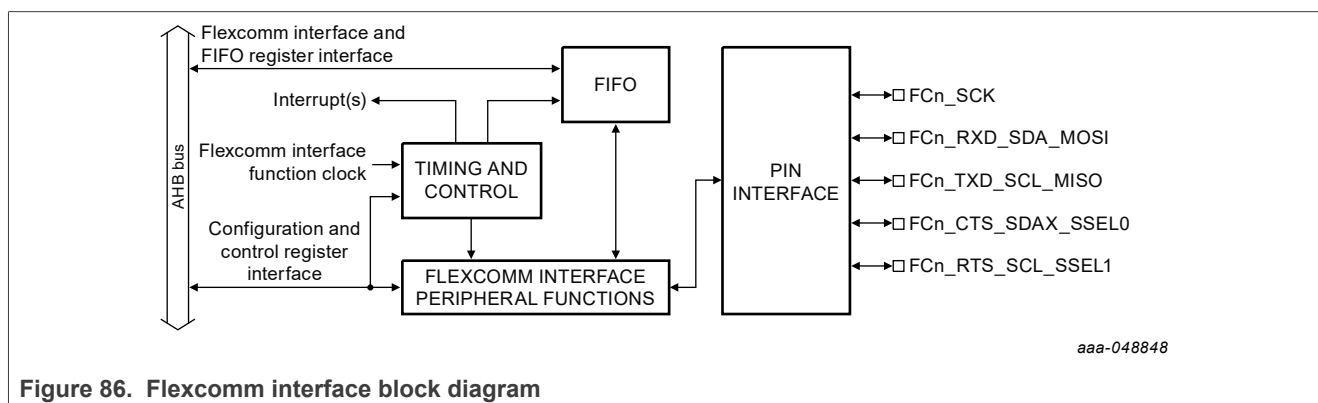
The Flexcomm interface is configured as follows:

1. Flexcomm interface clock: Select a clock source for the related Flexcomm interface. Options are shown in [Figure 3](#). Also see the sections "Flexcomm interface 0 clock source select for fractional rate divider (FCCLKSEL0)", "Flexcomm interface 1 clock source select for fractional rate divider (FCCLKSEL1)", and "Flexcomm interface 2 clock source select for fractional rate divider (FCCLKSEL2)" in the Register description ([Ref. 5](#)).
2. Select the required Flexcomm interface function by writing to the PSELID register of the related Flexcomm interface (see the section "Peripheral select and flexcomm ID register (PSELID)" in the Register description ([Ref. 5](#)).
3. See specific peripheral chapters for information on configuring those peripherals: [Section 34](#), [Section 35](#), and [Section 33](#).

Note: The Flexcomm interface (0 to 2) function clock frequency must not be higher than 32 MHz.

32.4 Architecture

[Figure 86](#) shows the overall structure of one Flexcomm.



32.4.1 Function summary

The MCX W23 devices incorporate Flexcomm interfaces and functions as shown in [Table 188](#). Specific part numbers and package variations may include a subset of this list.

Table 188. Flexcomm interface base addresses and functions

Flexcomm interface number	Base address	USART (see Section 34)	SPI (see Section 35)	I2C (see Section 33)
0	0x4008 6000	Yes	Yes	Yes
1	0x4008 7000	Yes	Yes	Yes
2	0x4008 8000	Yes	Yes	Yes

32.4.2 Choosing a peripheral function

Software writing to the PSELID register selects a specific peripheral function, from among the functions that a particular Flexcomm interface. Reading the PSELID register provides information on which peripheral functions are available on that Flexcomm interface.

When a specific peripheral function has been selected, the PID register supplies an identifier for the selected peripheral. Software may use this information to confirm the selection before proceeding.

32.4.3 FIFO usage

See the chapter for a specific peripheral function for information on how the FIFO is used (see [Section 32.4.1](#)).

32.4.4 DMA

The Flexcomm interface generates DMA requests if desired, based on a selectable FIFO level. See [Section 32.4.1](#) for a specific peripheral function for information on how the FIFO is used.

32.4.5 AHB bus access

Generally, the bus interface to the registers contained in the Flexcomm interface (including its serial peripheral functions) only supports word writes. Byte and halfword writes must 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. It allows support for control information embedded in DMA buffers, for example. (see the section "FIFO write data (FIFOWR)" in the Register description ([Ref. 5](#)).

32.5 Pin description

Each Flexcomm interface allows up to 5 pin connections. Specific uses of a Flexcomm interface typically do not use all of them. 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 189. 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.
RXD_SDA_MOSI	Input	Receive data input for the USART function.
	I/O	SDA (data) input/output for the I ² C function.
	I/O	Master data output/slave data input for the SPI function.
TXD_SCL_MISO	Output	Transmit data output for the USART function.
	I/O	SCL (clock) input/output for the I ² C function.
	I/O	Master data input/slave data output for the SPI function.
CTS_SDA_SSEL0	Input	Clear to send input for the USART function.
	I/O	SDA (data) input/output for the I ² C 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 ² C function.
	I/O	Slave select 1 input or output for the SPI function.

33 I²C-bus interfaces (I2C0 - I2C2)

I²C-bus functions are available on all MCX W23 devices as a selectable function in each Flexcomm interface. Up to 3 Flexcomm interfaces are available.

33.1 Features

- Independent master, slave, and monitor functions
- Bus speeds supported
 - Standard-mode, up to 100 kbits/s
 - Fast-mode, up to 400 kbits/s
 - Fast-mode Plus, up to 1 Mbps/s (on specific I²C pins)
- Supports both multimaster and multimaster with slave functions.
- Multiple I²C slave address supported in hardware.
- To respond to multiple I²C bus addresses, one slave address can be selectively qualified with a bit mask or an address range.
- 10-bit addressing supported with software assist.
- Supports system management bus (SMBus).
- Separate DMA requests for master, slave, and monitor functions.
- To receive and compare an address as a slave, no chip clocks are required. So, this event can wake up the device from deep-sleep mode.
- Automatic modes optionally allow less software overhead for some use cases.

33.2 Pin description

The I²C pins are fixed-pin functions and enabled through IOCON. See the IOCON settings in [Table 190](#) and in [Section 15](#).

Table 190. I²C-bus pin description

Function	Type	Pin name used in data sheet	Description
SCL	I/O	FCn_TXD_SCL_MISO or FCn_RTS_SCL_SSEL1	I ² C serial clock
SDA	I/O	FCn_RXD_SDA_MOSI or FCn_CTS_SDA_SSEL0	I ² C serial data

33.3 Basic configuration

Configure the I²C and related clocks as follows:

- If necessary, use the PRESETCTRL1 register (see the section "Peripheral reset control 1 (PRESETCTRL1)" in the Register description ([Ref. 5](#)) 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 the section "Flexcomm interface" in the Register description ([Ref. 5](#))). If the Flexcomm interface itself is reset via the PRESETCTRL1 register, any selection that has been made is cleared.
- Configure the I²C for the desired functions:
 - In the AHBCLKCTRL1 register (see the section "AHB clock control 1 (AHBCLKCTRL1)" in the Register description ([Ref. 5](#))), set the appropriate bit for the related Flexcomm interface 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 [Section 15](#). Configure the I²C clock and data rate. It includes the CLKDIV register for both master and slave modes, and MSTTIME for

master mode (see the section "Timeout value register (TIMEOUT)" in the Register description ([Ref. 5](#)) and [Section 33.5.2](#)).

Note: The Flexcomm interface function clock frequency must not exceed 32 MHz.

Note: While the I²C function is incorporated into the Flexcomm interface, it does not use the Flexcomm interface FIFO.

33.3.1 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 must be configured as required for the I²C-bus mode that is used (SM, FM, FM+, HS) via the IOCON (see the section "Digital I/O control for port 0 pins (PIO0_13 - PIO0_14)" in the Register description ([Ref. 5](#)).

If these pins or standard pins are used, they must be configured as described in IOCON section (see the sections "Digital I/O control for port 0 pins PIO0_b (PIO0_0 - PIO0_12)" and "Digital I/O control for port 0 pins PIO_b (PIO_15 - PIO_31)" in the Register description ([Ref. 5](#)).

The state of the MSTPENDING status bit controls the transmission of the address and data bits. When 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 allows the desired I²C-bus rate. Divide the clock as necessary (see the section "Clock predivider for the entire I2C interface (CLKDIV)" in the Register description ([Ref. 5](#)).
- Further divide the source clock, if necessary, using the CLKDIV register (see the section "Timeout value register (TIMEOUT)" in the Register description ([Ref. 5](#)).
- Set the SCL high and low times to complete the bus rate setup (see the section "Master control register (MSTCTL)" in the Register description ([Ref. 5](#)).

33.3.1.1 Master write to slave

Configure Flexcomm interface 1 as I²C interface (see the section "Flexcomm interface" in the Register description ([Ref. 5](#)).

Configure the I²C as a master: Set the MSTEN bit to 1 in the CFG register (see the section "Configuration for shared functions (CFG)" in the Register description ([Ref. 5](#)).

Write data to the slave:

1. Write the slave address with the $\overline{RW}$ bit set to 0 to the master data register MSTDAT (see the section "Slave control register (SLVCTL)" in the Register description ([Ref. 5](#)).
2. Start the transmission by setting the MSTSTART bit to 1 in the master control register (see the "Master control register (MSTCTL)" in the Register description ([Ref. 5](#)). The following happens:
 - The pending status is cleared and the I²C bus is busy.
 - The I²C master sends the start bit and address with the $\overline{RW}$ bit to the slave.
3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register
4. Write 8 bits of data to the MSTDAT register.
5. Continue with the transmission of data by setting the MSTCONT bit to 1 in the master control register (see the section "Master control register (MSTCTL)" in the Register description ([Ref. 5](#)). The following happens:
 - The pending status is cleared and the I²C-bus is busy.

- The I²C master sends the data bits to the slave address.
- Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
 - Stop the transmission by setting the MSTSTOP bit to 1 in the master control register (see the section "Master control register (MSTCTL)" in the Register description ([Ref. 5](#)).

Table 191. 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();
```

33.3.1.2 Master read from slave

Configure Flexcomm interface 1 as I²C interface (see the section "Flexcomm interface" in the Register description ([Ref. 5](#)).

Configure the I²C as a master: Set the MSTEN bit to 1 in the CFG register (see the section "Status register for master, slave, and monitor functions (STAT)" in the Register description ([Ref. 5](#)).

Read data from the slave:

- Write the slave address with the RW bit set to 1 to the master data register MSTDAT (see the section "Slave control register (SLVCTL)" in the Register description ([Ref. 5](#)).
- Start the transmission by setting the MSTSTART bit to 1 in the master control register (see the section "Master control register (MSTCTL)" in the Register description ([Ref. 5](#)). The following happens:
 - The pending status is cleared and the I2C bus is busy.
 - The I²C master sends the start bit and address with the RW bit to the slave.
- The slave sends eight bit of data.
- Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
- Read eight bits of data from the MSTDAT register.
- Stop the transmission by setting the MSTSTOP bit to 1 in the master control register (see the section "Master control register (MSTCTL)" in the Register description ([Ref. 5](#)).

Table 192. 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();
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
I2C->MSTDAT = (0x23 << 1) | 1; // address and 1 for R/W 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();
```

33.3.2 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.

If specialized I²C pins are used (PIO0_13 through PIO0_14), the pins must be configured as required for the I²C-bus mode that is used (SM, FM, FM+, HS) via the IOCON (see the section "Digital I/O control for port 0 pins (PIO_13 - PIO_14)" in the Register description ([Ref. 5](#)).

If these pins or standard pins are used, they must be configured as described in IOCON section (see the sections "Digital I/O control for port 0 pins PIO_b (PIO_0 - PIO_12)" and "Digital I/O control for port 0 pins PIO_b (PIO_15 - PIO_31)" in the Register description ([Ref. 5](#)).

The state of the SLVPENDING status bit controls the transmission of the address and data bits. When the status is slave pending, the slave can acknowledge (ack), 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.

33.3.2.1 Slave read from master

Configure Flexcomm interface 1 as I²C interface (see the section "Flexcomm interface" in the Register description ([Ref. 5](#))). Configure the I²C as a slave with address x:

Set the SLVEN bit to 1 in the CFG register (see the section "Status register for master, slave, and monitor functions (STAT)" in the Register description ([Ref. 5](#))).

- Write the slave address x to the address 0 match register (see the section "Slave control register (SLVCTL)" in the Register description ([Ref. 5](#))).

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 the section "Combined slave receiver and transmitter data register (SLVDAT)" in the Register description ([Ref. 5](#))).
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 the section "Combined slave receiver and transmitter data register (SLVDAT)" in the Register description ([Ref. 5](#)).

Acknowledge (ack) the data by setting SLVCONTINUE = 1 in the slave control register (see the section "Slave control register (SLVCTL)" in the Register description ([Ref. 5](#)).

Table 193. 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
```

33.3.2.2 Slave write to master

Configure Flexcomm interface 1 as I²C interface (see the section "Flexcomm interface" in the Register description ([Ref. 5](#)). Configure the I²C as a slave with address x:

- Set the SLVEN bit to 1 in the CFG register (see the section "Configuration for shared functions (CFG)" in the Register description ([Ref. 5](#)).
- Write the slave address x to the address 0 match register (see the section "Slave address register (SLVADR0)" in the Register description ([Ref. 5](#)).

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 the section "Slave control register (SLVCTL)" in the Register description ([Ref. 5](#)).
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 the section "Combined slave receiver and transmitter data register (SLVDAT)" in the Register description ([Ref. 5](#)).

Continue the transaction by setting SLVCONTINUE = 1 in the slave control register (see the section "Slave control register (SLVCTL)" in the Register description ([Ref. 5](#)).

Table 194. 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
```

33.3.3 Configure the I²C for wake-up

In sleep mode, any activity on the I²C bus that triggers an I²C interrupt can wake up the part, if the interrupt is enabled in the INTENSET register and the NVIC. As long as the Flexcomm interface clock remains active in sleep mode, the I²C can wake up the part independent of the I²C interface being configured in master mode or slave mode.

In deep-sleep mode, the I²C clock is turned off as are all peripheral clocks. However, if the I²C is configured in slave mode and an external master on the I²C bus provides the clock signal, the I²C interface can create an interrupt asynchronously. This interrupt, if enabled via the POWER_EnterDeepSleep() low-power API and in the I²C interface INTENCLR register, can then wake up the core.

33.3.3.1 Wake-up from sleep mode

- Enable the I²C interrupt in the NVIC.
- Enable the I²C wake-up event in the INTENSET register. Wake-up on any enabled interrupts is supported (see the INTENSET register). Examples are the following events:
 - Master pending
 - Change to idle state
 - Start/stop error
 - Slave pending
 - Address match (in slave mode)
 - Data available/ready

33.3.3.2 Wake-up from deep-sleep mode

- Enable the I²C interrupt in the NVIC.
- Enable the I²C interrupt using a 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.

33.4 General description

Figure 87 shows the architecture of the I²C-bus interface.

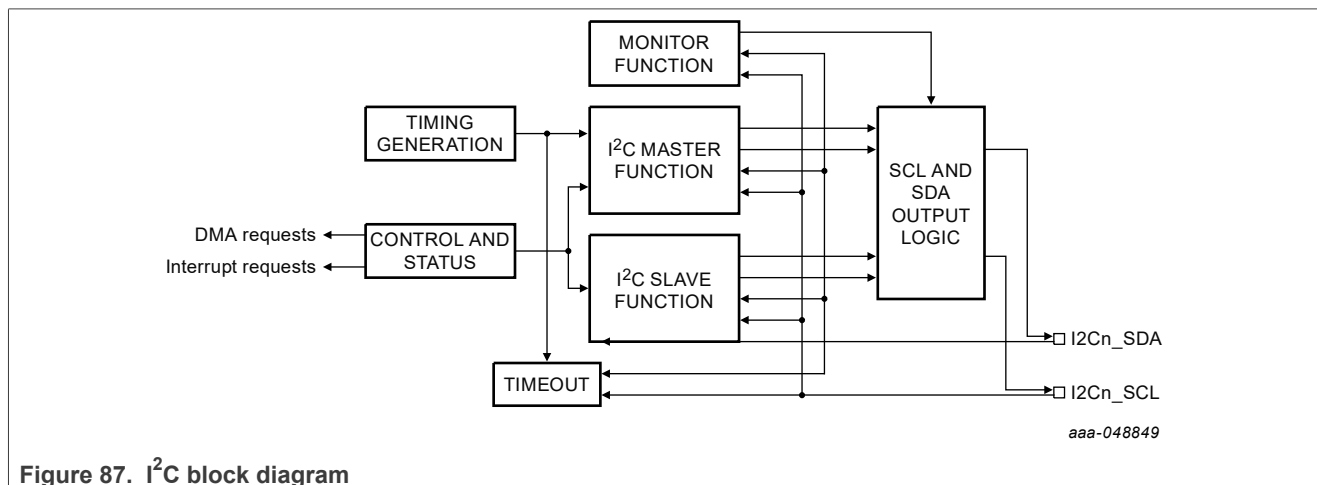


Figure 87. I²C block diagram

33.5 Functional description

33.5.1 AHB bus access

The bus interface to the I²C registers contained in the Flexcomm interface only supports word writes. Byte and halfword writes are not supported with the I²C function.

33.5.2 Bus rates and timing considerations

Due to the nature of the I²C bus, it is generally not possible to guarantee a specific clock rate on the SCL pin. On the I²C bus, the clock can be stretched by any slave device, extended by software overhead time, and so on.

In a multimaster system, the master that provides the shortest SCL high time causes that time to appear on SCL as long as that master participates in I²C traffic (that is, when it is the only master on the bus or during arbitration between masters).

In addition, I²C implementations generally base subsequent actions on what actually happens on the bus lines. For instance, a bus master allows SCL to go high. It then monitors the line to make sure that it actually did go high (requires in a multimaster system). The result is a small delay before the next action on the bus, caused by the rise time of the open-drain bus line.

Rate calculations give a base frequency that represents the fastest that the I²C bus can operate if nothing slows it down.

33.5.2.1 Rate calculations

Master timing

SCL high time (in Flexcomm Interface function clocks) = I²C clock divider * SCL high multiplier
(see the sections "Interrupt status register for master, slave, and monitor functions (INSTAT)" and "Combined master receiver and transmitter data register (MSTDAT)" in the Register description ([Ref. 5](#)).

Nominal SCL rate = Flexcomm Interface function clock rate / (SCL high time + SCL low time)

Note: DIVVAL must be 1.

Note: For a 400 kHz clock rate, the clock frequency after the I2C divider (divval) must be ≤ 2 MHz. [Table 196](#) shows the recommended settings for the 400 kHz clock rate.

Table 195. Settings for the 400 kHz clock rate

Input clock to I2C	DIVVAL for CLKDIV register	MSTSCLHIGH for MSTTIME register	MSTSCLOW for MSTTIME register
30 MHz	14	0	1
24 MHz	14	0	0

Slave timing

SCL received from the I²C bus master controls most aspects of slave operation. However, if the slave function stretches SCL to allow for software responses, it must provide sufficient data setup time to the master before releasing the stretched clock. It 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 must be configured such that one clock time is greater than the $t_{\text{SU,DAT}}$ value noted in the I²C bus specification for the I²C mode that is being used.

33.5.2.2 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)

The I²C interface supports standard mode, fast mode, and fast mode plus with the same software sequence, which also supports SMBus.

Clock stretching

When the clock does not have sufficient information on how to proceed, that is, software has not supplied data and/or instructions to generate a start or stop, the I²C interface automatically stretches it. In principle, at least, I²C can allow any bus participant at any time that SCL is low, in SM, FM, and FM+ modes to stretch the clock.

In practice, the I²C interface described here may stretch SCL at the following times, in SM, FM, and FM+ modes:

- As a slave:
 - After an address is received that complies with at least one slave address (before the address is acknowledged).
 - As a slave receiver, after each data byte received (software then acknowledges the data).
 - As a slave transmitter, after each data byte is sent and the matching acknowledge is received from the master.
- As a master:
 - After each address is sent and the acknowledge bit is received.
 - As a master receiver, 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 is received from the slave.

In each case, the relevant pending flag (MSTPENDING or SLVPENDING) is set at the point where clock stretching occurs.

33.5.3 Timeout

A timeout feature on an I²C interface can be used to detect a stuck bus and potentially do something to alleviate the condition. Two different types of timeout are supported. Both types apply when the I²C interface and the timeout function are enabled. Master, slave, or monitor functions do not have to be enabled.

In the first type of timeout, reflected by the EVENTTIMEOUT flag in the STAT register, the time between bus events governs the timeout check. These events include start, stop, and all changes on the I²C clock (SCL). When the time between any of these events is longer than the time configured in the TIMEOUT register, this timeout is asserted. If problems occur, it can be useful in monitoring an I²C bus within a system as part of a method to keep the bus running.

The SCLTIMEOUT flag in the STAT register reflects the second type of I²C timeout. When the SCL signal remains low longer than the time configured in the TIMEOUT register, this timeout is asserted. It corresponds with the SMBus timeout parameter T_{TIMEOUT}. In this situation, if a slave is the offending device, it can reset its own I²C interface. If all listening slaves (including masters that can be addressed as slaves) reset their own I²C interface, the bus is released. Unless it is a current master causing the problem. For more details, see the SMBus specification ([Ref. 10](#)).

Both types of timeout are generated only when the I²C bus is considered busy. That is, when a start condition occurred more recently than a stop condition.

33.5.4 Ten-bit addressing

Ten-bit addressing is accomplished by the I²C master sending a second address byte to extend a particular range of standard 7-bit addresses. If the master writes to the slave, the I²C frame simply continues with data after the two address bytes. For the master to read from a slave, it must 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 the previous write operation addressed it and stay selected for the subsequent read with the correct partial I²C address.

For the master function, the I²C is 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 it can be used to intercept just one (see the section "Slave address register (SLVADR1 - SLVADSR3)" in the Register description ([Ref. 5](#))). If in 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. If a follow-up read operation occurs, the slave function must record that it has been addressed.

For slave transmitter mode, the slave function responds to the initial address in the same fashion as for slave receiver mode. It 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 if the first part of the 10-bit address is a complete match to the previous address before acknowledging the address.

33.5.5 Clocking and power considerations

To operate, the master function of the I²C always requires a peripheral clock to be running. When the slave is not addressed, the slave function can operate without any internal clocking. It means that reduced power modes up to deep-sleep mode can be entered and, when the I²C slave function recognizes an address, the device wakes up. When information becomes available, the monitor mode can similarly wake up the device from a reduced power mode.

33.5.6 Interrupt handling

The I²C provides a single interrupt output that handles all interrupts for master, slave, and monitor functions.

33.5.7 DMA

DMA with the I²C is done only for data transfer. DMA cannot handle the control of the I²C. When DMA transfers data, I²C acknowledges are handled implicitly. No CPU intervention is required while DMA is transferring data.

Generally, the DMA can handle data transfers for master mode after a slave sends and acknowledges an address. For slave mode, the DMA can handle data transfers 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, the DMA can transfer receive data and transmit data. The DMA supports three DMA requests: data transfer in master mode, slave mode, and monitor mode.

To minimize software overhead time for I²C handling, DMA may be used with automatic operation.

A received NACK (from a slave in master mode or from a master in slave mode) causes the DMA to stop. An interrupt is generated. A repeated start sensed on the bus similarly causes DMA to stop. Then also, an interrupt is generated.

If a channel is available, the monitor function may be used with DMA. For more information on how DMA channels are used with the monitor function, see [Table 172](#).

33.5.7.1 DMA as a master transmitter

A basic sequence for a master transmitter:

1. Software sets up DMA to transmit a message.
2. Software causes a slave address with write command to be sent and checks that the address was acknowledged.
3. Software turns on DMA mode in the I²C.
4. DMA transfers data and eventually completes the transfer.
5. Software causes a stop (or repeated start) to be sent.

Software handles any exceptions to the standard transfer, such as the slave sending a NACK before the end of the transfer.

33.5.7.2 DMA as a master receiver

A basic sequence for a master receiver:

1. Software sets up DMA to receive a message.
2. Software causes a slave address with read command to be sent and checks that the address was acknowledged.
3. Software starts DMA.
4. DMA completes.
5. Software causes a stop or repeated start to be sent.

Software handles any exceptions to the standard transfer.

33.5.7.3 DMA as a slave transmitter

A basic sequence for a slave transmitter:

1. Software acknowledges an I²C address.
2. Software sets up DMA to transmit a message.
3. Software starts DMA.
4. DMA completes.

33.5.7.4 DMA as a slave receiver

A basic sequence for a slave receiver:

1. Software receives an interrupt for a slave address received and acknowledges the address.
2. Software sets up DMA to receive a message, less the final data byte.
3. Software starts DMA.
4. DMA completes.
5. Software sets SLVNACK before receiving the final data byte.
6. Software receives the final data byte.

33.5.8 Automatic operation

Automatic operation modes provide a way to reduce software overhead for I²C slave functions with some limitations. They are intended to be used primarily in conjunction with slave DMA. Related control bits are SLVDMA, AUTOACK, and AUTOMATCHREAD in the SLCCTL register and the AUTONACK bit in the SLVADR0 register. [Table 196](#) shows how these controls may be used. These cases apply when an address matching SLVADR0, qualified by SLVQUAL0, is received.

Table 196. 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.
x	1	Yes	Yes	ACK	Automatic slave DMA Expected read/write case. When the automatic ACK is sent, the SLVDMA bit is set and the AUTOACK bit is cleared.
1	0	x	No	NACK	Bus is ignored until software changes the setup.
1	1	No	No	NACK	Bus is ignored until software changes the setup.

34 Universal synchronous/asynchronous receiver/transmitter (USART0 - USART2)

USART functions are available on all MCX W23 devices as a selectable function in each Flexcomm interface peripheral. Up to three Flexcomm interfaces are available.

34.1 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 32.768 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 operate with the system DMA controller.
- Loopback mode for testing of data and flow control.

34.2 Basic configuration

Initial configuration of a USART peripheral is accomplished as follows:

- If necessary, use the PRESETCTRL1 register (see the sections "Peripheral reset control 1 (PRESETCTRL1)" and "Peripheral reset control 2 (PRESETCTRL2)" in the Register description ([Ref. 5](#)) 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 the section "Peripheral select and flexcomm ID register (PSELID)" in the Register description ([Ref. 5](#)).
- Configure the FIFOs for operation.
- Configure USART for receiving and transmitting data:
 - In the AHBCLKCTRL1 register (see the section "AHB clock control 1 (AHBCLKCTRL1)" in the Register description ([Ref. 5](#)), 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 [Section 15](#)).

- Configure the Flexcomm interface clock and USART baud rate (see [Section 34.2.1](#)).
Note: The Flexcomm interface function clock frequency is 10 MHz (sync mode) or 6.25 MHz (async) mode.
- Configure the USART to wake up the part from low-power modes (see [Section 34.2.2](#)).
- Configure the USART to receive and transmit data in synchronous slave mode (see [Section 34.2.2](#)).

34.2.1 Configure the Flexcomm interface clock and USART baud rate

Each Flexcomm interface has a separate clock selection, which can include a shared fractional divider (see [Section 34.5.2.3](#)). 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 required 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.

$$\text{Flexcomm interface clock} = (\text{FRG input clock}) / (1 + (\text{MULT} / \text{DIV}))$$

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.

For more information on the FRG, see the section "System control (SYSCON)" in the Register description ([Ref. 5](#)).

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 required to produce the desired baud rate.

Generally: $\text{baud rate} = [\text{FCLK} / \text{oversample rate}] / \text{BRG divide}$

With specific register values: $\text{Baud rate} = [\text{FCLK} / (\text{OSRVAL} + 1)] / (\text{BRGVAL} + 1)$

Generally: $\text{BRG divide} = [\text{FCLK} / \text{oversample rate}] / \text{baud rate}$

With specific register values: $\text{BRGVAL} = [[\text{FCLK} / (\text{OSRVAL} + 1)] / \text{baud rate}] - 1$

See the section "USART" in the Register description ([Ref. 5](#)).

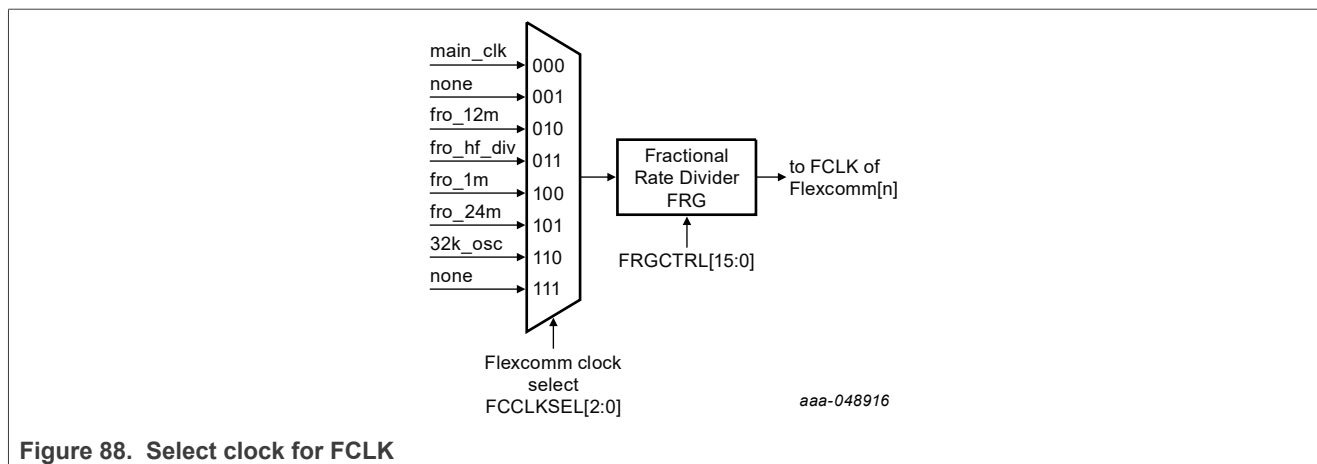
3. In synchronous master mode: The serial clock is $\text{Un\_SCLK} = \text{FCLK} / (\text{BRGVAL} + 1)$.

The 32.768 kHz RTC oscillator can also clock the USART. Set the MODE32K bit to enable this 32.768 kHz mode (see [Section 34.5.2.3](#)).

To ensure that synchronous UART mode works at 25 MHz in slave receive mode. The UCLK duty cycle must be 50 %.

If an FRG clock is used to derive the clock from the source clock, the 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. So, if a 25 MHz interface frequency is targeted, use a method other than FRG to derive the interface clock. You can select the 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 does not have a 50 % duty cycle. So, if the FRG is used, the BRG must also be set to divide the clock by some integer factor before it is used.



For details on the clock configuration, see [Section 34.5.2](#).

34.2.2 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 32.768 kHz mode is enabled, the USART can run in asynchronous mode using the 32.768 kHz RTC oscillator and create interrupts.

34.2.2.1 Wake-up from sleep mode

- Configure the USART in either asynchronous mode or synchronous mode (see the section "USART" in the Register description ([Ref. 5](#))).
- Enable the USART interrupt in the NVIC.
- Any enabled USART interrupt wakes up the part from sleep mode.

34.2.2.2 Wake-up from deep-sleep mode

1. Configure the USART in synchronous slave mode (see the section "USART" in the Register description ([Ref. 5](#))). The SCLK function must be connected to a pin and also connect the pin to the master. Alternatively, the 32.768 kHz mode can be enabled and the USART operated in asynchronous mode with the 32.768 kHz RTC oscillator.
2. Enable the USART interrupt using low-power API
3. Enable the USART interrupt in the NVIC.
4. 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.

Note: By enabling or disabling specific USART interrupts, you can customize when the wake-up occurs.

34.3 Pin description

The USART receive, transmit, and control signals are movable Flexcomm interface functions. They are assigned to external pins via IOCON (see [Section 15](#)) to assign the USART functions to pins on the device package. [Table 198](#) shows the recommended IOCON settings.

Table 197. USART pin description

Pin	Type	Name used in pin configuration chapter	Description
TXD	O	FCn_TXD_SCL_MISO	Transmitter output for USART on Flexcomm interface n. Serial transmit data.
RXD	I	FCn_RXD_SDA_MOSI	Receiver input for USART on Flexcomm interface n. Serial receive data.
RTS	O	FCn_RTS_SCL_SSEL1	Request-to-send output for USART on Flexcomm interface n. This signal supports interprocessor communication by using 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.
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 the CTSEn bit in the CFG register enables it and when it is configured to appear on a device pin. When the external device deasserts (high) the USART, it completes transmitting any character already in progress, then stops 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. Note: When the USART is configured as a master, such that SCK is an output, it must actually be connected to a pin for the USART to work properly.

Table 198. Suggested USART pin settings

IOCON bits	Type D pin (PIO0_0 - PIO0_12 and PIO0_15 - PIO0_19)	Type I pin (PIO0_13 or PIO0_14)
31:16	Reserved	Reserved
15	Reserved	I2CFILTER: Configures I2C features for standard mode, fast mode, and fast mode plus operation. 0b - Enabled, I2C 50 ns glitch filter enabled 1b - Disabled, I2C 50 ns glitch filter disabled Set to 1 to disable I2C filter
14	Reserved	EGP: Select GPIO / I ² C mode. Set to 1 for GPIO mode
13	Reserved	ECS: Pull-up current source enable in IIC mode. 0b - Enabled, pull resistor is connected 1b - Disabled, IO is in open-drain Set to 0. It does not have any effect in GPIO mode
12	Reserved	FILTEROFF: Controls input glitch filter for I2C. 0 for fast or standard mode I2C 1 for fast mode plus Set to 1.
9	OD: Controls open-drain mode. 0b - Normal, normal push-pull output 1b - Open-drain, Simulated open-drain output (high drive disabled) Set to 0 unless open-drain is desired.	Same as type D.
8	DIGIMODE: 0b - Analog mode, digital input is disabled 1b - Digital mode, digital input is enabled Set to 1.	Same as type D.
7	INVERT: Input polarity. 0b - Disabled, input function is not inverted 1b - Enabled, input is function inverted Set to 0.	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 Set to 0.	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 Set to 0.	Same as type D.

Table 198. Suggested USART pin settings...continued

IOCON bits	Type D pin (PIO0_0 - PIO0_12 and PIO0_15 - PIO0_19)	Type I pin (PIO0_13 or PIO0_14)
3:0	FUNC: Selects pin function. 0001b - Alternative connection 1 0010b - Alternative connection 2	Same as type D.
General comment	A good choice for USART input or output	Not recommended for USART functions that can be outputs in the chosen mode.

34.4 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 reception, they are passed to the receiver FIFO to await access by the CPU or DMA controller.

The USART transmitter block accepts data that the CPU or DMA controller has written 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 32.768 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.

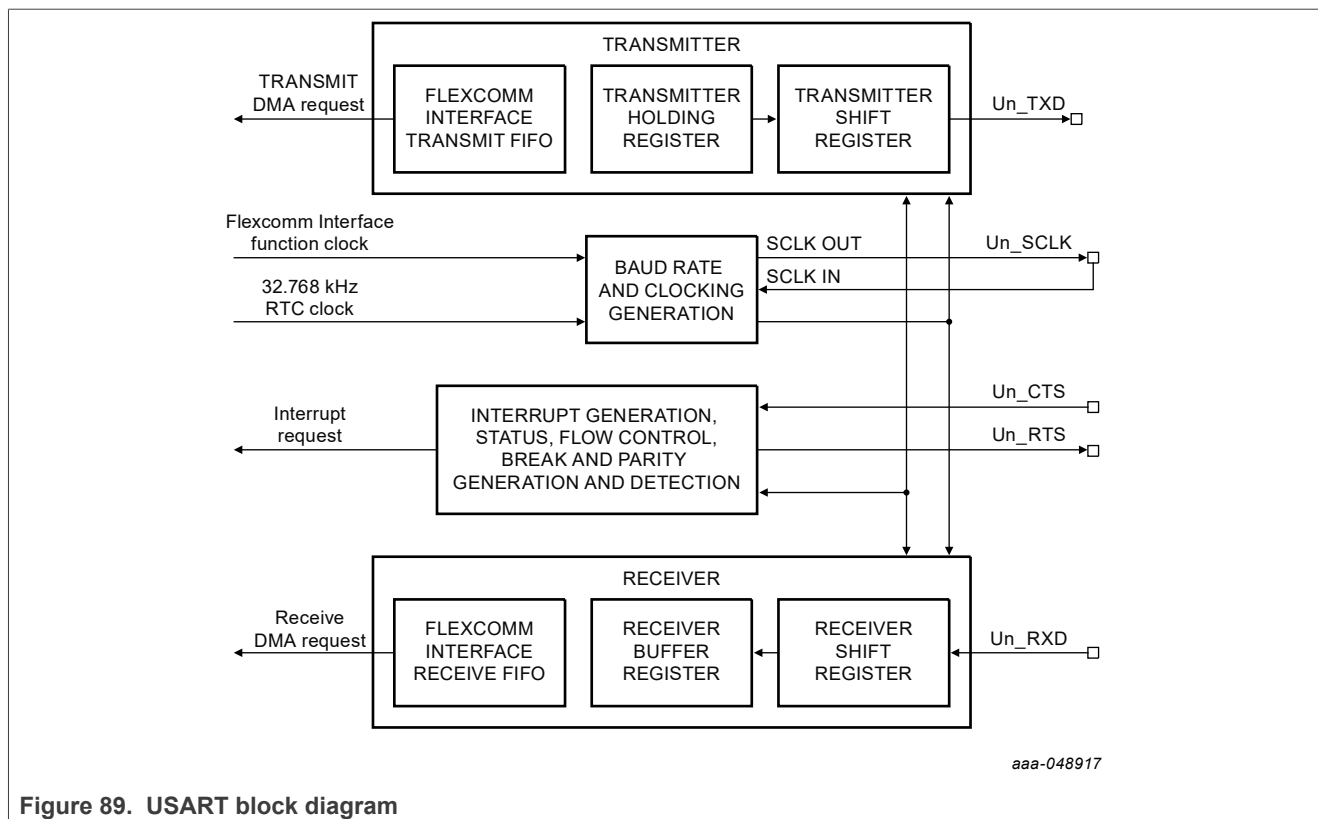


Figure 89. USART block diagram

34.5 Functional description

34.5.1 AHB bus access

The bus interface to the USART registers contained in the Flexcomm interface only support word writes. Byte and halfword writes are not supported in the USART function.

34.5.2 Clocking and baud rates

To use the USART, clocking details must be defined such as setting up the clock source selection, the BRG, and the FRG if it is the selected clock source.

Also see [Section 34.2.1](#).

34.5.2.1 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 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 required.

The FRG register controls the FRG, which provides the base clock that any Flexcomm interface may use. The FRG creates a lower rate output clock by suppressing selected input clocks. When not required, the value of 0 can be set for the FRG. The input clock is not divided then.

The FRG output clock is defined as the input clock divided by $1 + (\text{MULT} / 256)$, where MULT is in the range of 1 to 255. It allows the production of 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). The integer BRG divider contained in each USART can divide any further division, specific to each USART block.

The base clock that the FRG produces cannot be perfectly symmetrical. So, the FRG distributes the output clocks as evenly as is practical. Since USARTs normally use 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 the sections "Fractional rate divider for flexcomm 0 (FLEXFRG0CTRL)", "Fractional rate divider for flexcomm 1 (FLEXFRG1VTRL)", and "Fractional rate divider for flexcomm 2 (FLEXFRG2CTRL)" in the Register description ([Ref. 5](#)).

34.5.2.2 Baud rate generator (BRG)

The baud rate generator (see the section "Baud rate generator register (BRG)" in the Register description ([Ref. 5](#))) is used to divide the base clock to produce a rate of 16 times the desired baud rate. Integer divisions of higher baud rates can generate the standard baud rates.

34.5.2.3 32.768 kHz mode

To use a 32.768 kHz clock to operate a USART at any reasonable speed, a number of adaptations must be made. First, 16x overclocking must be abandoned. Otherwise, the maximum data rate is very low. For the same reason, multiple samples of each data bit must be reduced to one. Finally, special clocking must be used for individual bit times because 32.768 kHz is not close to an integer of any standard baud rate.

When 32.768 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. The rate error is less than 0.5 % in this mode, provided the RTC oscillator operates at the intended frequency of 32.768 kHz.

34.5.3 DMA

A DMA request is provided for each USART direction. It can be used instead of interrupts for transferring data by configuring the DMA controller and FIFO level triggering appropriately. The DMA controller provides an acknowledgment signal that, when it completes handling a related request, clears that request. When the transmitter can accept more data, the transmitter DMA request is asserted. When received data is available to be read, the receiver DMA request is asserted.

When DMA is used to perform USART data transfers, other mechanisms can be used to generate interrupts, if necessary. 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.

34.5.4 Synchronous mode

In synchronous mode, a master generates a clock as the clock selection and BRG defines. The generated clock is used to transmit and receive data. As a slave, the external clock is used to transmit and receive data. No overclocking in either case.

34.5.5 Flow control

The USART supports hardware and software flow control.

34.5.5.1 Hardware flow control

The USART supports hardware flow control using RTS and/or CTS signaling. If RTS is configured to appear on a device pin so that it can be sent to an external device, it indicates to an external device that the receiver

can 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. Internal and external CTS can be used separately or together.

Figure 90 shows an overview of RTS and CTS within the USART.

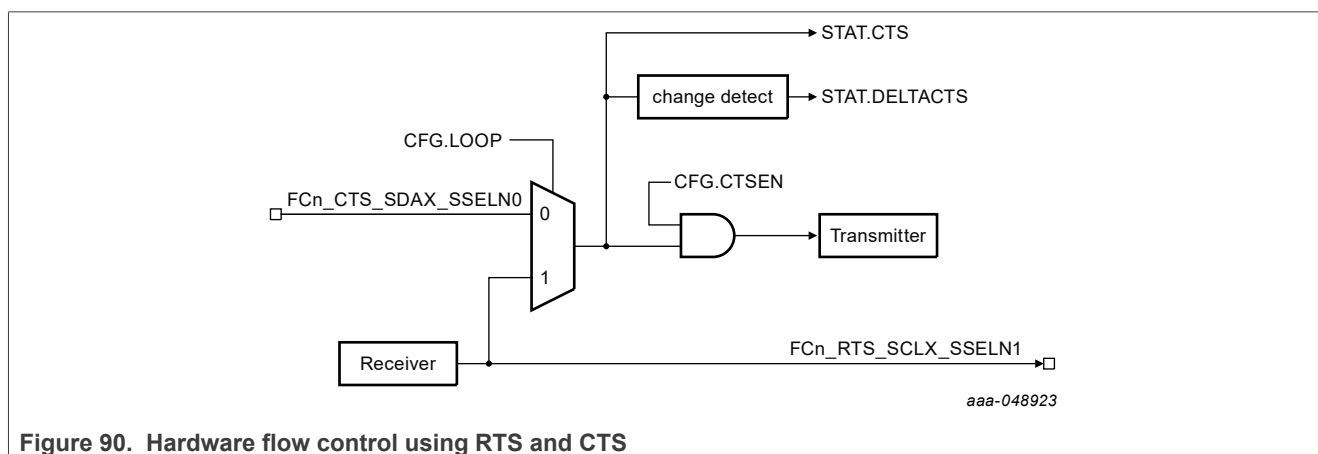


Figure 90. Hardware flow control using RTS and CTS

34.5.5.2 Software flow control

Software flow control can include XON/XOFF flow control or other mechanisms. 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) supported these mechanisms. The ability of software to turn off the transmitter gracefully (via the TXDIS bit in the CTL register) also supports these mechanisms.

34.5.6 Autobaud function

The autobaud functions attempts to measure the start-bit time of the next received character. For the measurement to work, the measured character must have a 1 in the least significant bit position, so that a falling and rising edge bounds the start bit. Before an autobaud operation is requested, the BRG value must be set to 0. The measurement is done 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 timeout that ABERR flags if no character is received at the expected time. It is recommended that autobaud is only enabled when the USART receiver is idle. When enabled, either data become available in the FIFO or ABERR is asserted at some point, at which time software must turn off autobaud.

When the USART is in synchronous mode, autobaud has no meaning and must not be enabled.

34.5.7 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 section "USART configuration register (CFG)" and the ADDR bit in the section "Address register for automatic address matching (ADDR)" in the Register description ([Ref. 5](#)). It also provides software address recognition (see the ADDRDET bit in the section "USART control register (CTL)" in the Register description ([Ref. 5](#)).

Automatic data direction control with the RTS pin can be set up using the OESEL, OEPOL, and OETA bits in the CFG register (see the section "USART configuration register (CFG)" in the Register description ([Ref. 5](#)). Data direction control can also be implemented in software using a GPIO pin.

34.5.8 Oversampling

Typical industry standard USARTs use a 16x oversample clock to transmit and receive asynchronous data. It 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. No oversampling occurs in synchronous modes.

When the baud rate is very high or the function clock is very low, reducing the oversampling can help getting a better baud rate matching. 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 %.

Altering the oversampling has consequences. 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, one oversample clock, then three data samples is taken. Then, before the end of the bit time, one more oversample clock is taken. Since the oversample clock runs asynchronously from the input data, skew of the input data relative to the expected timing has little room for error. At 16x oversampling, several oversample clocks are available before actual data sampling is done, making the sampling more robust. Use the highest oversampling where the rate error is acceptable in the system.

34.5.9 Break generation and detection

A line break may be sent at any time, regardless of other USART activity. A received break is also detected at any time, including during reception of a character. When the RX input remains low for 16 bit times, the received break is signaled. The DELTARXBRK status flag, which can be used as an interrupt notes the beginning and the end of a received break. See [Section 34.5.10](#) for details of LIN mode break.

To avoid that any character that is transmitted is corrupted, disable the USART transmitter by setting the TXDIS bit in the CTL register. Then, wait for the TXDISSTAT flag to be set before a break is sent. Then, a 1 may be written to the TXBRKEN bit in the CTL register. It sends a break until TXBRKEN is cleared, allowing the sending of breaks of any lengths.

34.5.10 LIN bus

The only difference between standard operation and LIN mode is that LIN mode alters the way break generation and detection are performed (see [Section 34.5.9](#) for details). When a break is requested by setting the TXBRKEN bit in the CTL register, and then sending a dummy character, a 13-bit time break is sent. When the RX input remains low for 11 bit times, a received break is flagged. For non-LIN mode, a received character is also flagged a framing error status accompanies it.

As a LIN slave, the autobaud feature can be used to synchronize to a LIN sync byte. It returns the value of the synchronous byte as confirmation of success.

Wake-up for LIN can 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 receives internal clocks that allow it to function, it can be set to wake up the CPU for any interrupt, including a received start bit. If no clocks are running, the GPIO function of the USART RX pin can be programmed to wake up the device.

35 Serial peripheral interfaces (SPI0 - SPI2)

SPI functions are available on all MCX W23 devices as a selectable function in each Flexcomm interface. Up to three Flexcomm interfaces are available.

35.1 Features

- Master and slave operation.
- Data transmits of 4 to 16 bits supported directly. Software supports larger frames.
- The SPI function supports separate transmit and receive FIFOs with eight entries each.
- Supports DMA transfers: SPIn transmit and receive functions can operate with the system DMA controller.
- Data can be transmitted to a slave without the necessity to read incoming data. This option can be useful while setting up an SPI memory.
- Up to two slaves select input/outputs with selectable polarity and flexible usage.

35.2 Basic configuration

Initial configuration of an SPI peripheral is accomplished as follows:

- If necessary, use the PRESETCTRL1 register (see the section "Peripheral reset control 1 (PRESETCTRL1)" in the Register description ([Ref. 5](#)) to reset the Flexcomm interface that has a specific peripheral function selected.
- Select the desired Flexcomm interface function by writing to the PSELID register of the related Flexcomm interface (see the section "Peripheral select and flexcomm ID register (PSELID)" in the Register description ([Ref. 5](#)).
- Configure the FIFOs for operation.
- Configure the SPI for receiving and transmitting data
 - In the AHBCLKCTRL1 (see the section "AHB clock control 1 (AHBCLKCTRL1)" in the Register description ([Ref. 5](#)), 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 (see the section "Main clock B source select (MAINCLKSELB)" in the Register description ([Ref. 5](#)).
 - Configure the required Flexcomm interface pin functions through IOCON (see [Section 35.3](#)).
 - Configure the Flexcomm interface clock and SPI data rate (see [Section 35.5.4](#)).
 - Set the RXIGNORE bit to only transmit data and not read the incoming data. Otherwise, when the FIFORD buffer is full, the transmit halts.
 - For a slave, set the TXIGNORE bit to only receive data.

The Flexcomm interface function clock frequency must not exceed 32 MHz in master mode. In slave mode, it must not exceed 18 MHz for a VDD_IO voltage below 1.8 V. If the VDD_IO voltage is 1.8V or above, the maximum clock frequency in slave mode is 24 MHz.

- Configure the SPI function to wake up the part from low-power modes (see [Section 35.2.1](#)).

35.2.1 Configure the SPI for wake-up

In sleep-mode, any signal that triggers an SPI interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the SPI clock is configured to be active in sleep-mode, the SPI can wake up the part independently of whether the SPI block is configured in master or slave mode.

In deep-sleep mode, the SPI clock is turned off. However, if the SPI is configured in slave mode and an external master on the bus provides the clock signal, the SPI can create an interrupt asynchronously and wake up the device. The appropriate interrupts must be enabled in the SPI and the NVIC.

35.2.1.1 Wake-up from sleep mode

- Configure the SPI in master or slave mode (see the section "SPI configuration register (CFG)" in the Register description ([Ref. 5](#)).
- Enable the SPI interrupt in the NVIC.
- Any enabled SPI interrupt wakes up the part from sleep mode.

35.2.1.2 Wake-up from deep-sleep mode

- Configure the SPI in slave mode (see the section "SPI configuration register (CFG)" in the Register description ([Ref. 5](#)). The SCK function must be connected to a pin and the pin connected to the master
- Enable the SPI interrupt as wake-up source using the POWER_EnterDeepSleep low-power API.
- Enable the SPI interrupt in the NVIC.
- Enable desired SPI interrupts. Examples are the following wake-up events:
 - A change in the state of the SSEL pins
 - Data available to be received
 - Receive FIFO overflow

35.3 Pin description

The SPI signals are movable Flexcomm interface functions. They are assigned to external pins via IOCON (see [Section 15](#)). [Table 200](#) shows the recommended IOCON settings.

Table 199. SPI pin description

Function	Type	Pin name used in pin description chapter	Description
SCK	I/O	FCn_SCK	Serial clock for SPI on Flexcomm interface n. SCK is a clock signal used to synchronize the transfer of data. The master drives it. The slave receives it. 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 when the master bit in CFG equals 1, regardless of the state of the enable bit.
MOSI	I/O	FCn_RXD_SDA_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 when the master bit in CFG equals 1, regardless of the state of the enable bit.

Table 199. SPI pin description...continued

Function	Type	Pin name used in pin description chapter	Description
MISO	I/O	FCn_TXD_SCL_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 one of the SSL signals selects the slave.
SSEL0	I/O	FCn_CTS_SDA_SSEL0	Slave select 0 for SPI on Flexcomm interface n. When the SPI interface is a master, it drives the SSEL signals to an active state before the start of serial data. It then releases them to an inactive state after the serial data has been sent. By default, this signal is active-low. However, it 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 addressed. The SSEL pin is driven when the master bit in the CFG register equals 1, regardless of the state of the enable bit.
SSEL1	I/O	FCn_RTS_SCL_SSEL1	Slave select 1 for SPI on Flexcomm interface n.

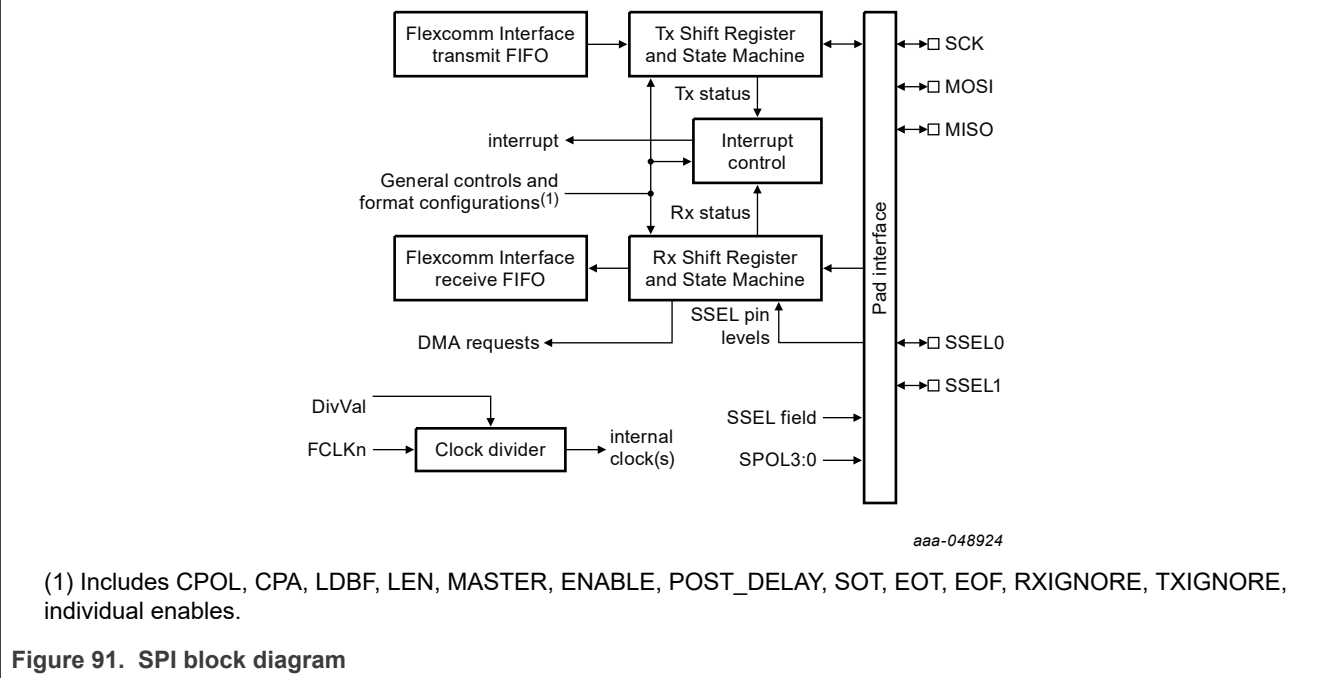
Table 200. Suggested SPI pin settings

IOCON bits	Type D pin (PIO0_0 - PIO0_12 and PIO0_15 - PIO0_19)	Type I pin (PIO0_13 or PIO0_14)
31:16	Reserved	Reserved
15	Reserved	I2CFILTER Configures I2C features for standard mode, fast mode, and fast mode plus operation. 0b - Enabled, I2C 50 ns glitch filter enabled 1b - Disabled, I2C 50 ns glitch filter disabled Set to 1 to disable I2C filter
14	Reserved	EGP; select GPIO / I ² C mode. Set to 0 for I2C mode.
13	Reserved	ECS Pull-up current source enable in IIC mode. 0b - Enabled, pull resistor is connected. 1b - Disabled, IO is in open-drain. Generally set to 0. It does not have any effect in GPIO mode
12	Reserved	FILTEROFF Controls input glitch filter for I2C. 0 for fast or standard mode I2C 1 for fast mode plus Generally set to 1.

Table 200. Suggested SPI pin settings...continued

IOCON bits	Type D pin (PIO0_0 - PIO0_12 and PIO0_15 - PIO0_19)	Type I pin (PIO0_13 or PIO0_14)
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
8	DIGIMODE: . 0b - Analog mode, digital input is disabled 1b - Digital mode, digital input is enabled. Generally set to 1.	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
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
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
3:0	FUNC: Selects pin function. 0001b - Alternative connection 1 0010b - Alternative connection 2	Same as type D

35.4 General description



35.5 Functional description

35.5.1 AHB bus access

Except for 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 the SPI function for those registers.

To allow writing FIFO data without affecting the SPI control fields above bit 15 (see the section "FIFO write data (FIFOWR)" in the Register description (Ref. 5), the FIFOWR register also supports byte and half-word (data only) writes.

35.5.2 Operating modes: clock and phase selection

SPI interfaces typically allow configuration of clock phase and polarity. They are sometimes referred to as numbered SPI modes, as described in Table 201 and shown in Figure 92. Bits in the CFG register configure CPOL and CPHA (see the section "SPI configuration register (CFG)" in the Register description (Ref. 5).

Table 201. 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

Table 201. SPI mode summary...continued

CPOL	CPHA	SPI Mode	Description	SCK rest state	SCK data change edge	SCK data sample edge
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

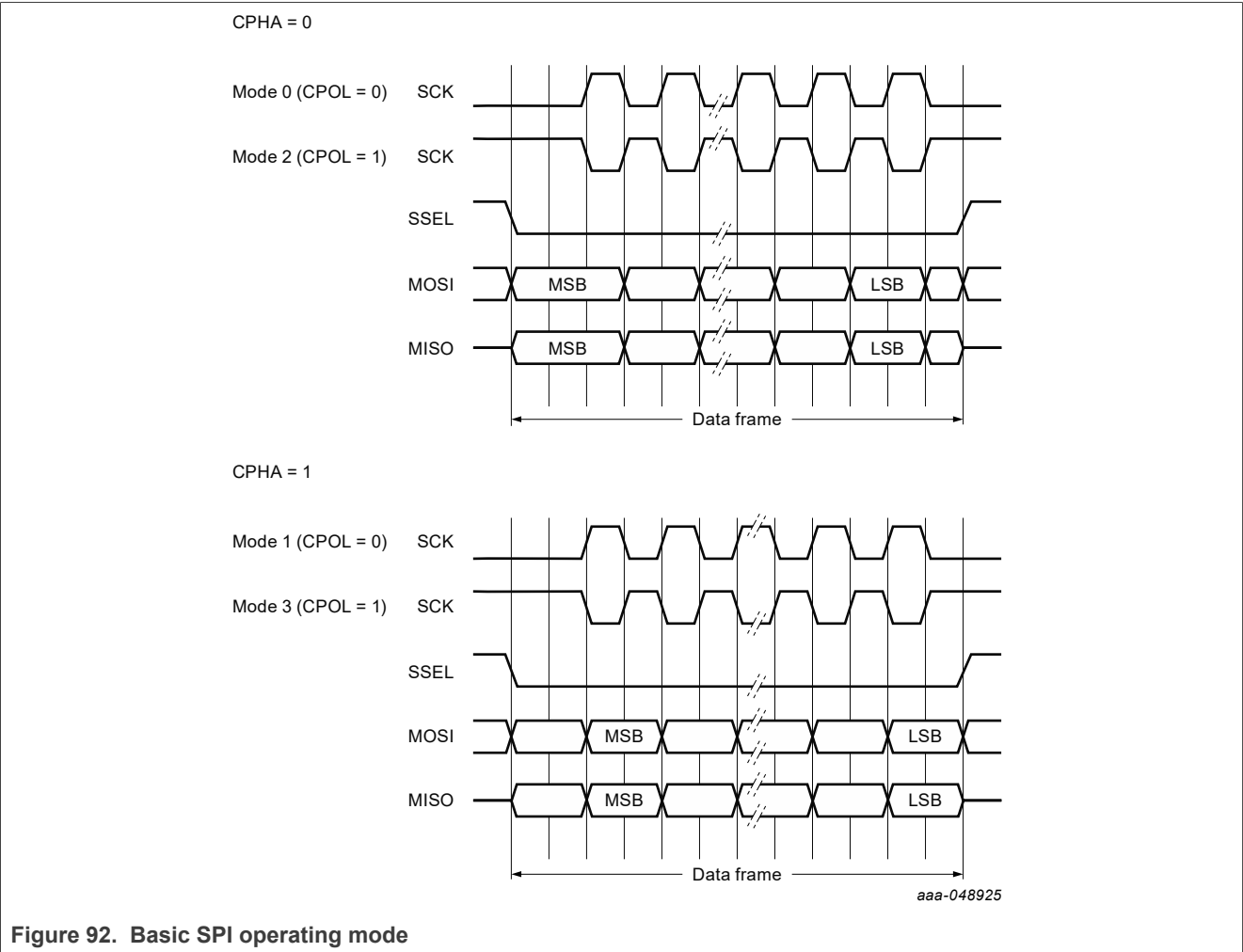


Figure 92. Basic SPI operating mode

35.5.3 Frame delays

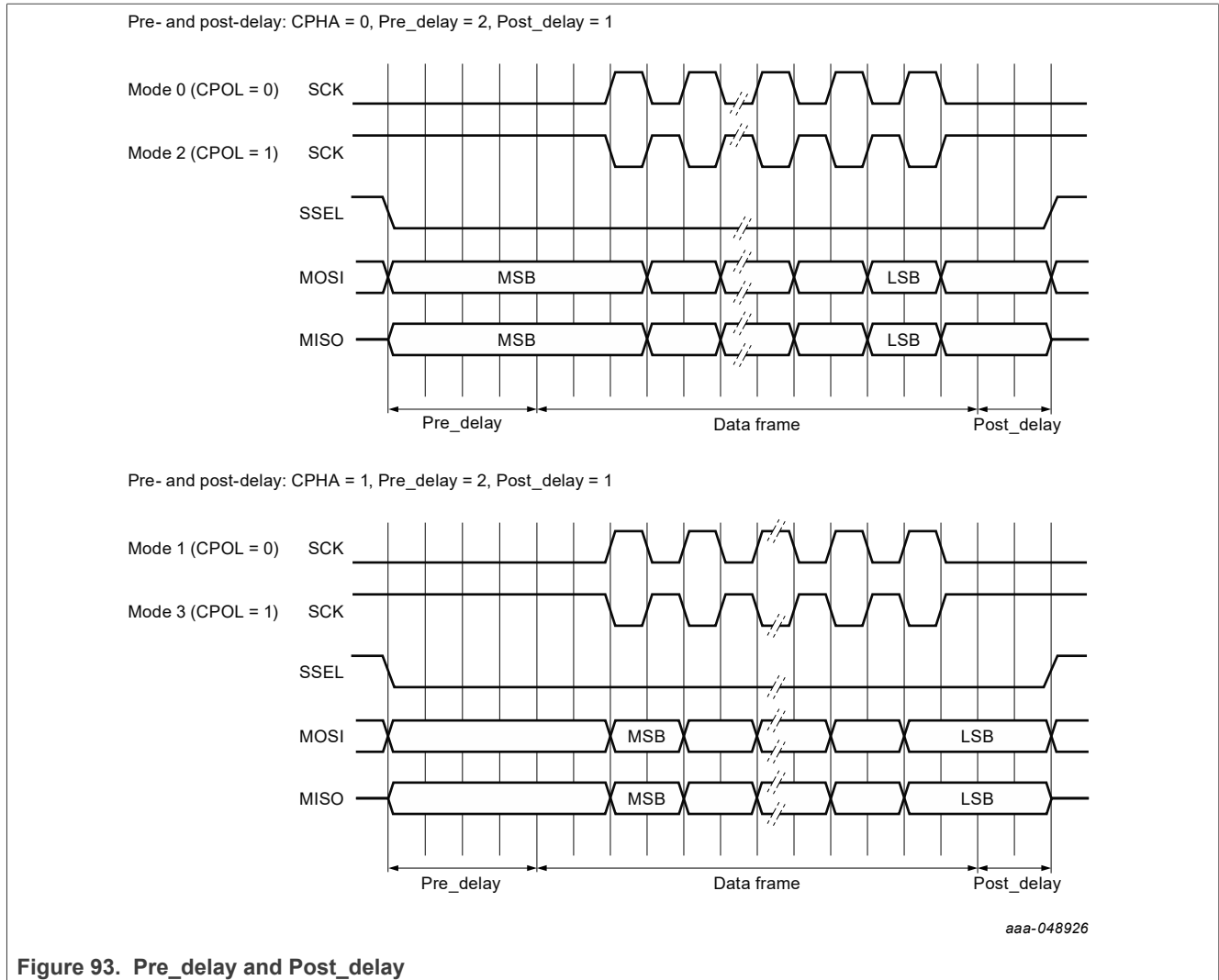
Several delays can be specified for SPI frames:

- Pre_delay:
Delay after SSEL is asserted before data clocking begins.
- Post_delay
Delay at the end of a data frame before SSEL is deasserted.
- Frame_delay
Delay between data frames when SSEL is not deasserted.

- Transfer_delay
Minimum duration of SSEL in the deasserted state between transfers.

35.5.3.1 Pre_delay and Post_delay

The examples in [Figure 93](#) show the Pre_delay and the Post_delay. The Pre_delay value controls the time between the assertion of SSEL and the beginning of the subsequent data frame. The Post_delay value controls the time between the end of a data frame and the deassertion of SSEL.



35.5.3.2 Frame_delay

The Frame_delay value controls the time at the end of each frame. This delay is inserted when the EOF bit = 1. The examples in [Figure 94](#) show the Frame_delay. Frame boundaries only occur where specified. The reason is that frame lengths can be of any size, involving multiple data writes (see [Section 35.5.7](#) for more information).

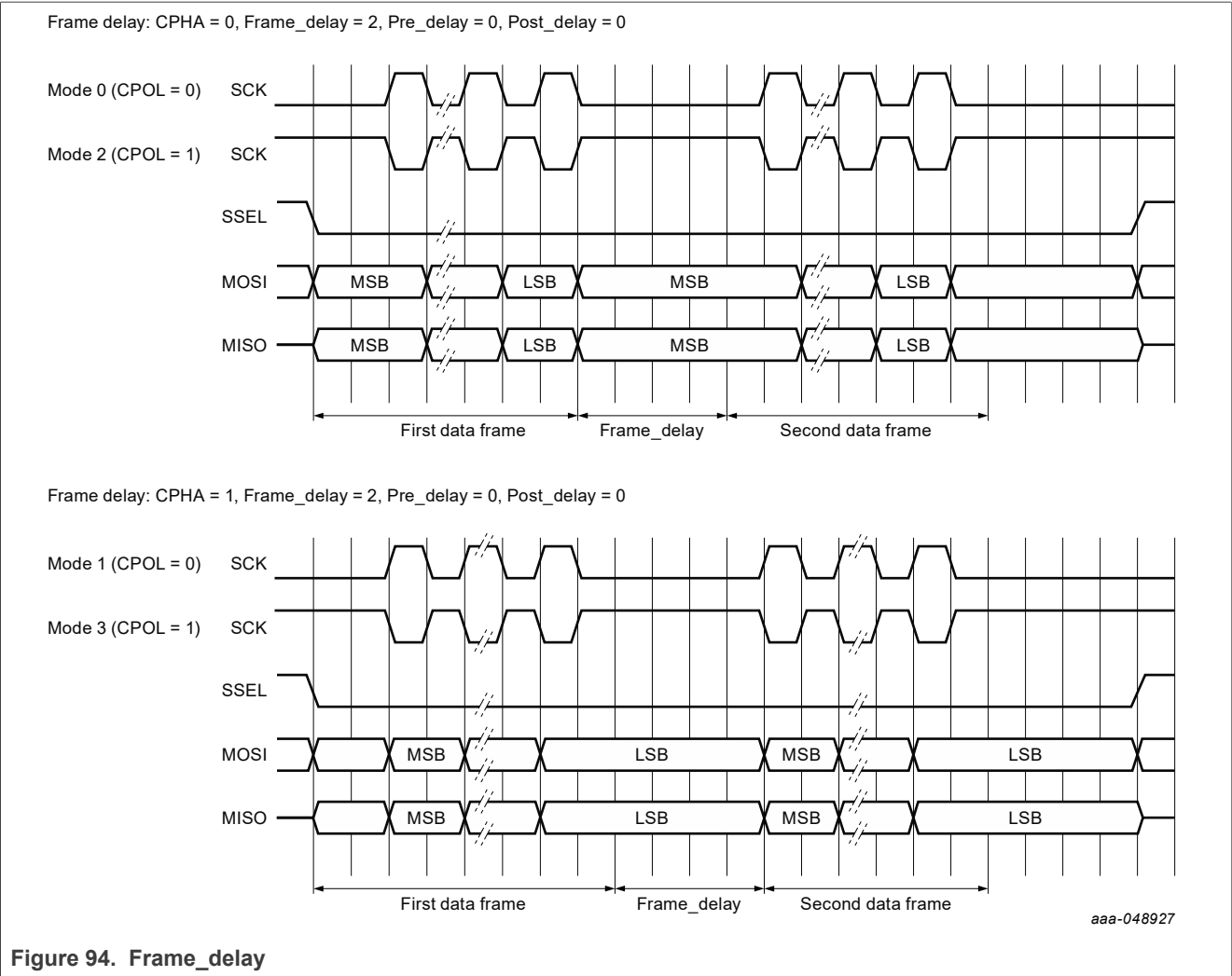


Figure 94. Frame_delay

35.5.3.3 Transfer_delay

The Transfer_delay value controls the minimum 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. The examples in [Figure 95](#) show the Transfer_delay.

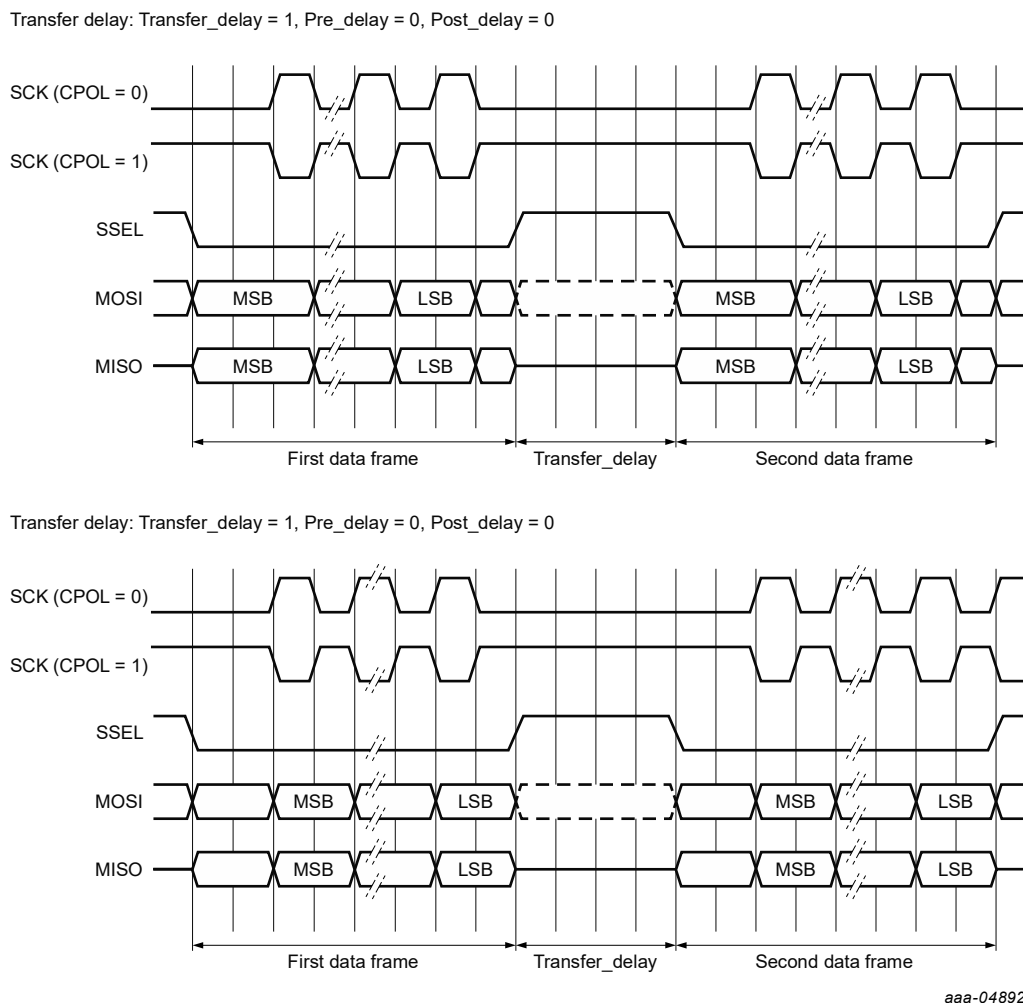


Figure 95. Transfer_delay

35.5.4 Clocking and data rates

To use the SPI, clocking details must be defined. They include configuring the system clock and selection of the clock divider value in DIV (see [Figure 3](#)).

35.5.4.1 Data rate calculations

The SPI interface is designed to operate asynchronously from any on-chip clocks and without the requirement for overclocking.

In slave mode, it 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 that the SPI clock divider produces 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.

35.5.5 Slave select

The SPI block provides for two slaves 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 2 SSELs in a register is always active-low. If an SSEL is inverted, it is done as the signal leaves/enters the SPI block.

In slave mode, any asserted SSEL that is connected to a pin activates the SPI. In master mode, all SSELs that are connected to a pin are output as defined in the SPI registers. In the latter case, the SSELs can be decoded externally in order to address more than two slave devices.

Note: When data is transferred in master mode, at least one SSEL is asserted.

In master mode, slave selects come from the TXSSEL bits in the FIFOWR register. In slave mode, the state of the two SSELs, along with received data in the RXSSEL_N field of the FIFORD register, are saved.

35.5.6 DMA operation

A DMA request is provided for each SPI direction. It can be used instead of interrupts for transferring data by configuring the DMA controller appropriately. The DMA controller provides an acknowledgment signal that, when it completes handling a related request, clears that request.

When TX DMA is enabled and the transmitter can accept more data, the transmitter DMA request is asserted.

When RX DMA is enabled and received data is available to be read, the receiver DMA request is asserted.

35.5.6.1 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 is set 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

- Using DMA interrupt and a second DMA transfer
To use only 8 bit 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 is 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.
- Using DMA and SPI interrupts (or background SPI status polling)
To use only one 8 bit or 16 bit wide DMA transfer for all the data, two interrupts are required to terminate the transfer properly (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.
To force termination after all data output is complete, the interrupt function handling the SPI TXLVL sets the SPI STAT register "END TRANSFER" bit.
- Using DMA linked descriptor
The DMA controller provides a linked list of DMA transfer control descriptors. The initial descriptors can be used to transfer all but the last data byte/half-word. These data transfers can be done as 8 bit 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 include the setting of the EOT bit.
Note: The DMA interrupt function cannot set the SPI status register (STAT) END TRANSFER control bit. It may terminate the transfer while the FIFO still has data to send.
- 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. It also allows a series of SPI transactions involving multiple slaves with one DMA operation, by changing the TXSSELn_N bits.

35.5.7 Data lengths greater than 16 bits

The SPI interface handles data frame sizes from 4 bits 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, one of 16 bits and one of 8 bits or two groups of 12 bits, and so on. Frames of any size, including greater than 32 bits, can be supported in the same way.

Details of how to handle larger data widths depend on other SPI configuration options. For instance, if slave selects must be deasserted between frames, it must be suppressed when a larger frame is split into more than one part. Sending two groups of 12 bits with SSEL deasserted between 24-bit increments, for instance, requires changing the value of the EOF bit on alternate 12-bit frames.

35.5.8 Data stalls

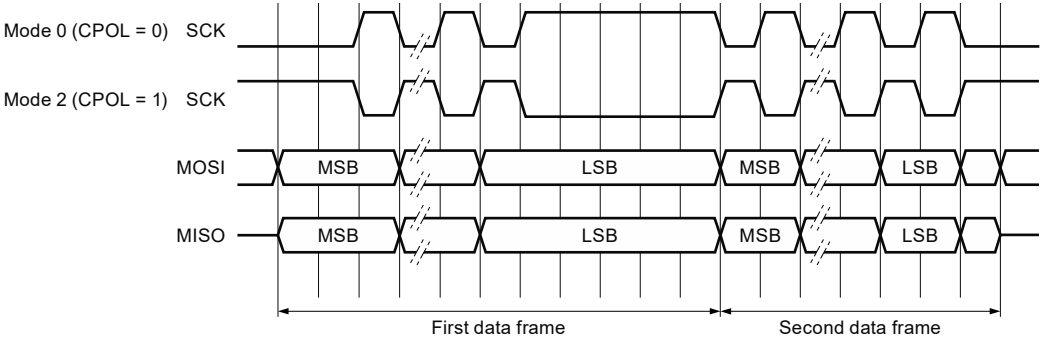
A stall for master transmit data can happen in modes 0 and 2 when SCK cannot be returned to the rest state until the MSB of the next data frame can be driven on MOSI. In this case, the stall happens just before the final clock edge of data if the next piece of data is not yet available.

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, stalls occur if the FIFO is full when the next piece of data is received. This stall happens one clock edge earlier than the transmitter stall.

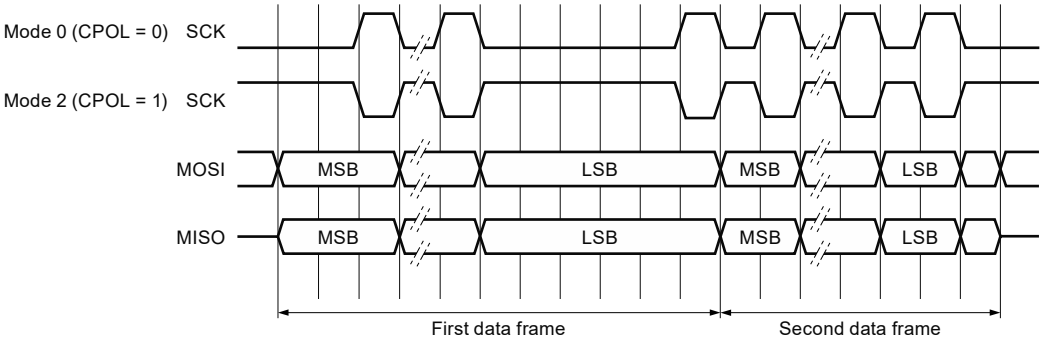
In modes 1 and 3, the same receiver stall can occur, but just before the final clock edge of the received data. Also, a transmitter stall does not happen in modes 1 and 3 because the transmitted data is complete at the point where a stall otherwise occurs. So, it is not necessary.

Stalls are reflected in the STAT register by the stalled status flag, which indicates the current SPI status. The transmitter is stalled until data is read from the receive FIFO. Use the RXIGNORE control bit setting to avoid the need to read the received data.

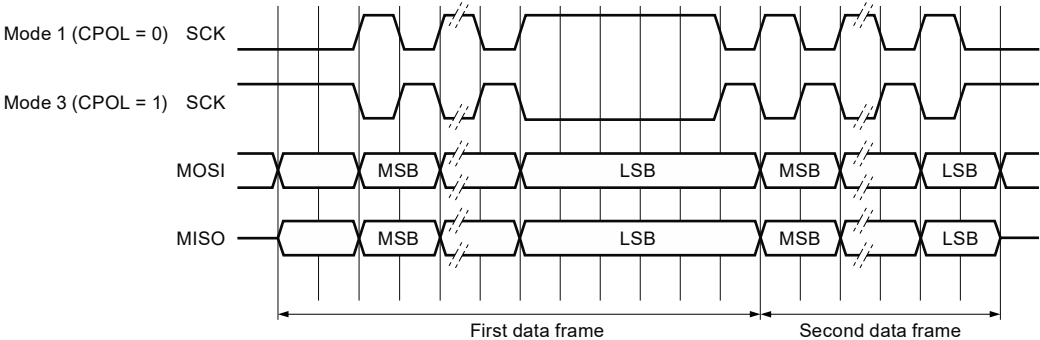
Transmitter stall: CPHA = 0, Frame_delay = 0, Pre_delay = 0, Post_delay = 0,2 clock stall



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



aaa-048929

Figure 96. Examples of data stalls

36 SPI flash interface (SPIFI)

The SPI flash interface is available on all MCX W23 devices.

36.1 Features

- Quad SPI flash interface (SPIFI) interface to external flash
- Transfer rates of up to SPIFI_CLK/2 bytes per second
- Code in the serial flash memory can be executed as if it was in the internal memory space of the CPU. It is accomplished by mapping the external flash memory directly into the CPU memory space.
- Supports 1-bit, 2-bit, and 4-bit bidirectional serial protocols
- Half-duplex protocol compatible with various vendors and devices

36.2 Basic configuration

Initial configuration of the SPIFI peripheral is accomplished as follows:

1. Peripheral clock: Set bit SPIFI in the AHBCLKCTRL0 register (see the section "AHB clock control 0 (AHBCLKCTRL0)" in the Register description ([Ref. 5](#)).
2. SPIFI clock: set up the SPIFI clock using the SPIFICKSEL and SPIFICKDIV registers (see [Figure 3](#)). Also see the Flexcomm interface clock select registers (sections "Flexcomm interface 0 clock source select for fractional rate divider (FCCLKSEL0)", "Flexcomm interface 1 clock source select for fractional rate divider (FCCCLKSEL1)", and "Flexcomm interface 2 clock source select for fractional rate divider (FCCCLKSEL2)") in the Register description ([Ref. 5](#)).
3. Pins: Select SPIFI pins and pin modes through the relevant IOCON registers (see the section "Peripheral select and flexcomm ID register (PSELID)" in the Register description ([Ref. 5](#)).

36.3 General description

The SPI flash interface (SPIFI) allows low-cost serial flash memories to be connected to the CPU with little performance penalty compared to parallel flash devices with higher pin count.

Many serial flash devices use a half-duplex command-driven SPI protocol for device setup and initialization. Quad devices use a half-duplex, command-driven 4-bit protocol for normal operation. Different serial flash vendors and devices accept or require different commands and command formats. SPIFI provides sufficient flexibility to be compatible with common flash devices. It includes extensions to ensure compatibility with future devices.

Serial flash devices respond to commands that software sends or that the SPIFI automatically sends when software reads either of the two read-only serial flash regions in the memory map (see [Table 202](#)).

Table 202. SPIFI flash memory map

Memory	Address
SPIFI data	0x2800 0000 to 0x2FFF FFFF Note: The address space allocated to the SPIFI for direct execution. The area allocated allows a maximum of 128 MB of SPI flash to be mapped into the CPU memory space. In practice, the usable space is limited to the size of the connected device.

36.4 Pin description

Table 203. SPIFI pin description

Pin	Type	Description
SPIFI_SCK	O	Serial clock for the flash memory, switched only during active bits on the MOSI/IO0, MISO/IO1, and IO3:2 lines.
SPIFI_CSN	O	Chip select for the flash memory, driven low while a command is in progress, and high between commands. In the typical case of one serial slave, this signal can be connected directly to the device. If more than one serial slave is connected, software and off-chip hardware should use general-purpose I/O signals in combination with this signal to generate the chip selects for the various slaves.
SPIFI_IO0 / SPIFI_MOSI	I/O	This is an output except in quad/dual input data fields. After a quad/dual input data field, it becomes an output again one serial clock period after CS goes high.
SPIFI_IO1 / SPIFI_MISO	I/O	This is an output in quad/dual opcode, address, intermediate, and output data fields, and an input in SPI mode and in quad/dual input data fields. After an input data field in quad/dual mode, it becomes an output again one serial clock period after CS goes high.
SPIFI_IO[3:2]	I/O	These are outputs in quad opcode, address, intermediate, and output data fields, and inputs in quad input data fields. If the flash memory does not have quad capability, these pins can be assigned to GPIO or other functions.

36.5 SPIFI hardware

The SPIFI has a base address for the registers and a base address for the memory area in which the serial flash connected to the SPIFI can be read.

The first operation with the serial flash is to read JEDEC ID, which is implemented by most serial flash devices. Depending on the device identity code that the serial flash returns in this operation, device-specific commands are used for further operation. Programming and other operations on the serial flash can be performed using the register interface.

36.6 Supported devices

Multiple QSPI devices from various vendors can be used with the SPIFI interface.

36.7 Functional description

36.7.1 Data transfer

Serial SPI uses the signals SPIFI_SCK, SPIFI_CSN, SPIFI_MISO, and SPIFI_MOSI, while quad mode adds the two IO signals SPIFI_IO[3:2].

The SPIFI implements basic, dual, and quad SPI in half-duplex mode, in which the SPIFI always sends a command to a serial flash memory at the start of each frame. A frame is the sequence of bytes transmitted during one period with CS LOW. In general, commands start with an opcode byte, although some serial flashes allow a no-opcode mode in which commands start with the address to be read. In write commands, the SPIFI sends all data in the frame. While in read commands, the SPIFI sends the command. Then, the serial flash sends data to the SPIFI.

Classic SPI includes four modes (mode 0 to mode 3), of which the SPIFI and most serial flashes implement modes 0 and 3. In mode 0, the SCK line is LOW between frames. In mode 3, it is HIGH. In mode 0, the SPIFI drives the first data bits from the time that it drives CS LOW. It drives the rest of the data on falling edges of

SCK. In mode 3, the SPIFI drives SCK LOW one-half clock period after it drives CS LOW. It drives data on the falling edge of SCK. In either mode, the serial flash samples the data on the rising edges of SCK.

The same scheme (transmitter changes data on falling edges of SCK, receiver samples data on rising edges) is maintained for the entire frame, including read data that the serial flash sends to the SPIFI.

The SPI protocol avoids all issues of setup and hold times between the clock and data lines by using half of the SCK period to transmit the data. For high clock speeds, it is necessary to sample read data using a feedback clock. The FBCLK bit enables the feedback clock from the SCK pad sampling method. It provides the best possible timing margin for both read and write data under the opposite-edge scheme.

The maximizing clock frequency is of such importance that further improvement is sometimes necessary. This further improvement is achieved with using the whole serial clock period to transmit data. This choice is enabled for read data by setting the RFCLK bit. When this bit is 1, the SPIFI samples data on the falling edge of the serial clock that follows the rising edge which is normally used. RFCLK and FBCLK and MODE3 should not all be 1 because there is no falling edge of the feedback clock to capture the last bit of a frame.

Consult the data sheet of the serial flash device to be used for the formats of the commands that it supports. [Figure 97](#) shows commands consisting of only an opcode field, sent in SPI and quad modes. All fields are multiples of 8 bits long. Bytes are sent with the most significant bit first in SPI mode. They are sent with the most significant 4 bits first in quad mode.

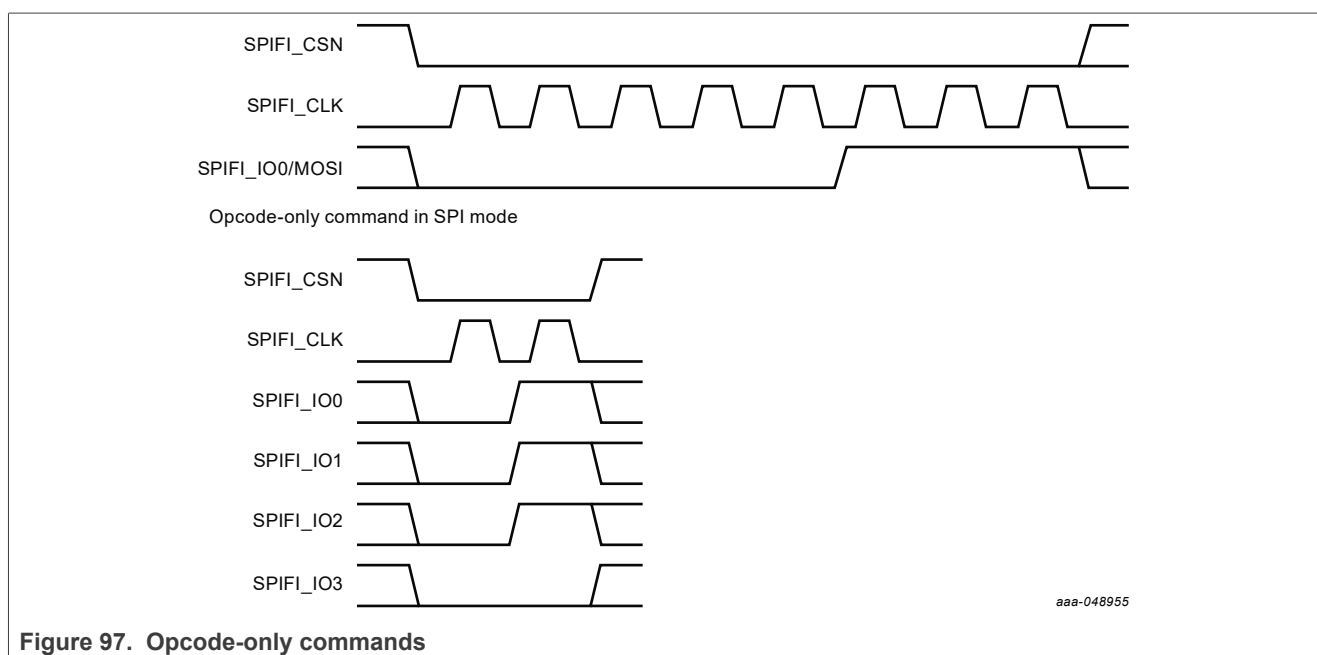
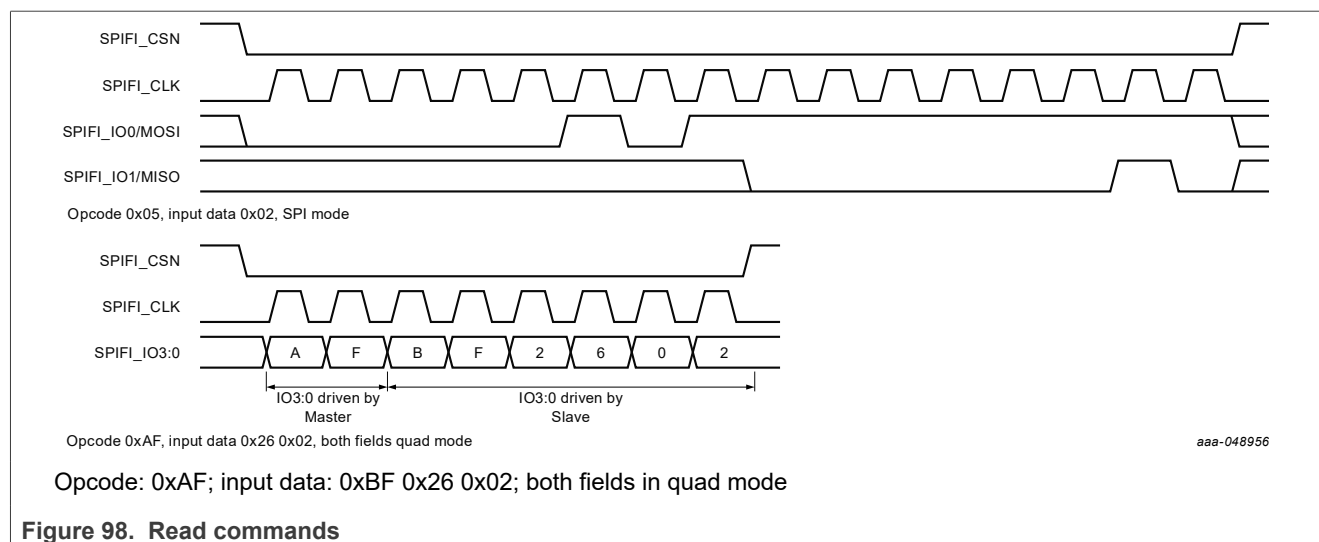


Figure 97. Opcode-only commands

[Figure 98](#) shows a command that reads 1 byte from the slave in SPI mode and a command that reads 3 bytes from the slave with the opcode and input data fields both in quad mode.

In quad mode, the SPIFI drives the IO3:0 lines during opcode, address, intermediate and output data phases. During the input data phase, the flash memory drives the IO3:0 lines. During address phase, the more significant bytes are sent first.



36.7.2 Software requirements and capabilities

During device setup, software must initialize the external serial flash device using those commands that place it in its highest-performance mode. When this sequence is complete, software must write the command that is issued in response to a read from the serial flash region of the memory map to the memory command register. If software attempts to read the flash region after reset, power-up, or writing the command register without writing the memory command register thereafter, the SPIFI responds with an abort error.

After writing the memory command register, the contents of the flash appear to the system as memory-mapped, which enables data access or execution from the serial flash over AHB.

SPIFI has two operational modes:

- **Memory mode**
The contents of the FLASH are memory-mapped in the chip.
- **Command mode**
The user can manually construct command sequences for the flash.

SPIFI cannot switch over from memory mode to command mode and vice versa without writing 1 to the RESET bit in the SPIFI status register and polling until hardware clears it to ensure that the current mode has been aborted.

Because the SPIFI is an AHB device, software or a DMA channel can read bytes, halfwords, or words from the flash region.

When necessary, deasserting HREADY delays reads from the flash region until the requested bytes are available to be read.

In memory mode, SPIFI prefetches sequential addresses to improve performance.

If no AHB accesses have taken place for a period specified by the timeout (TO) field in the control register, the SPIFI deasserts CS. When a new access that requires a new fetch of data occurs, the SPIFI reasserts CS and sends a new command to fetch the required data. It is done to save power in the SPI flash device.

If software reads or writes more data from the data register than was configured in the DATALEN field of the command register or reads or writes when no command was issued, the SPIFI hardware issues an abort exception.

When the serial flash must be programmed or erased, software must not write to the flash region of the address map. Instead, it must write the appropriate sequence of commands to the command, address, and data

registers. When an actual erase or program operation is under way in the serial flash device, software must write a read status command (with the POLL bit set) to the command register. Then:

- If INTEN in the control register is 1, the SPIFI interrupts the processor when the erase or write operation (and so the read status command) completes.
- If not, software can continually or periodically read the status register until it indicates that the read status command is complete.

When erasing or programming completes, software can do further programming or erasing or return to normal (memory mode) operation.

36.7.3 Peripheral mode DMA operation

To maintain synchronization between core accesses and serial data transfer with the serial flash, the SPIFI inserts wait states, during core read and write operations, when necessary. This mechanism is all that is required for load and store accesses and for memory-to-memory transfers by a DMA channel.

The peripheral mode is a mode that supports DMA transfers in which the SPIFI acts as a peripheral and drives a request signal to the DMA channel to control data transfer. This mode does not necessarily move data faster than memory-to-memory operation. However, it may be advantageous in systems in which software controls dynamic transfer of code and/or data between the serial flash and RAM on an as-required basis. The advantage is that clock cycles are not lost to wait states. So, the overall operation of the AHB is more efficient.

To have a burst size of one transfer, the DMA controller must be programmed to present word operations at the fixed address of the data register. The SPIFI drives the DMA request to the DMA controller.

To use this mode, software must write the command register to start the command and program a DMA channel, as described above, to transfer data between the data register and RAM. The SPIFI asserts the DMA request when:

- DMAEN in the control register is 1.
- MCINIT is 0.
- At least 4 bytes in the FIFO are available for a read operation. Or at least 4 empty byte locations are available in the FIFO for a write/program operation.

37 System controller (SYSCTL)

The system controller provides auxiliary control of the flash subsystem and the true random number generator. It also provides a gray to binary decoder.

37.1 Features

- Flash auxiliary control:
 - Flash stall enable/disable
 - Flash cache enable/disable
 - Flash speculation buffer enable/disable
- Gray to binary decoder:
Allows decoding gray values coming from OS event timer.
- True random number generator auxiliary control and status.

38 Programmable logical unit (PLU)

The PLU is available on all MCX W23 parts.

38.1 Features

- The programmable logic unit (PLU) is used to create small combinatorial and/or sequential logic networks, including simple state machines.
- The PLU consists 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.
- If a sequential network is implemented, an external clock to drive the four flip-flops must be applied to the PLU_CLKIN pin.
- Programmable logic can be used to drive on-chip inputs/triggers through external pin-to-pin connections.
- To facilitate programming of the PLU to implement the logic network described in a Verilog RTL design, a tool suite is provided.

38.2 Pin description

Up to six primary inputs into the PLU module, one clock input, and eight primary outputs are available. All the inputs are connected directly to the package pins via chip-level I/O multiplexing. Configuring the relevant IOCON registers can enable all these pins.

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 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, providing an input clock to PLU_CLKIN is not necessary.

Table 204. Time interval register (INTVAL[0:3], offset = 0x000 (INTVAL0) to 0x030 (INTVAL3))

Bit	Description	Pin
PLU_IN [5:0]	Primary inputs All plu_inputs are available as input sources to all the LUT elements.	PIO0_[10:5]
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.	PIO0_2 or PIO0_11
PLU_OUT [7:0]	Primary outputs Selectable via multiplexers from among all LUT element outputs and the four "state" flip-flops.	PIO0_[19:12]

38.3 General description

The PLU consists 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 consisting of primary inputs from pins, the outputs from all of the other LUTs, and the outputs of the four "state" flip-flops. To select the five inputs to

that LUT, a set of multiplexers associated with each LUT is used. Registers, which are programmed during initialization, control these multiplexers. Connecting multiple LUT elements together allows the 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. They can then be used as primary outputs and/or connected to the inputs of other LUTs. The external `plu_clkin` that the user supplies clocks these four flip-flops, if the target logic network uses them.

Note: *The internal chip reset signal automatically clears the four "state" flip-flops to '0'. 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 achieves 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.

Note: *When the PLU module is configured, the PLU bus clock can be shut off to conserve power. The only exception is when reading the outputs register while the PLU is operational is necessary. In that case, the PLU bus clock must be reenabled before performing the read.*

[Figure 99](#) shows the PLU block diagram with the following configuration: 26 LUT elements; six primary inputs.

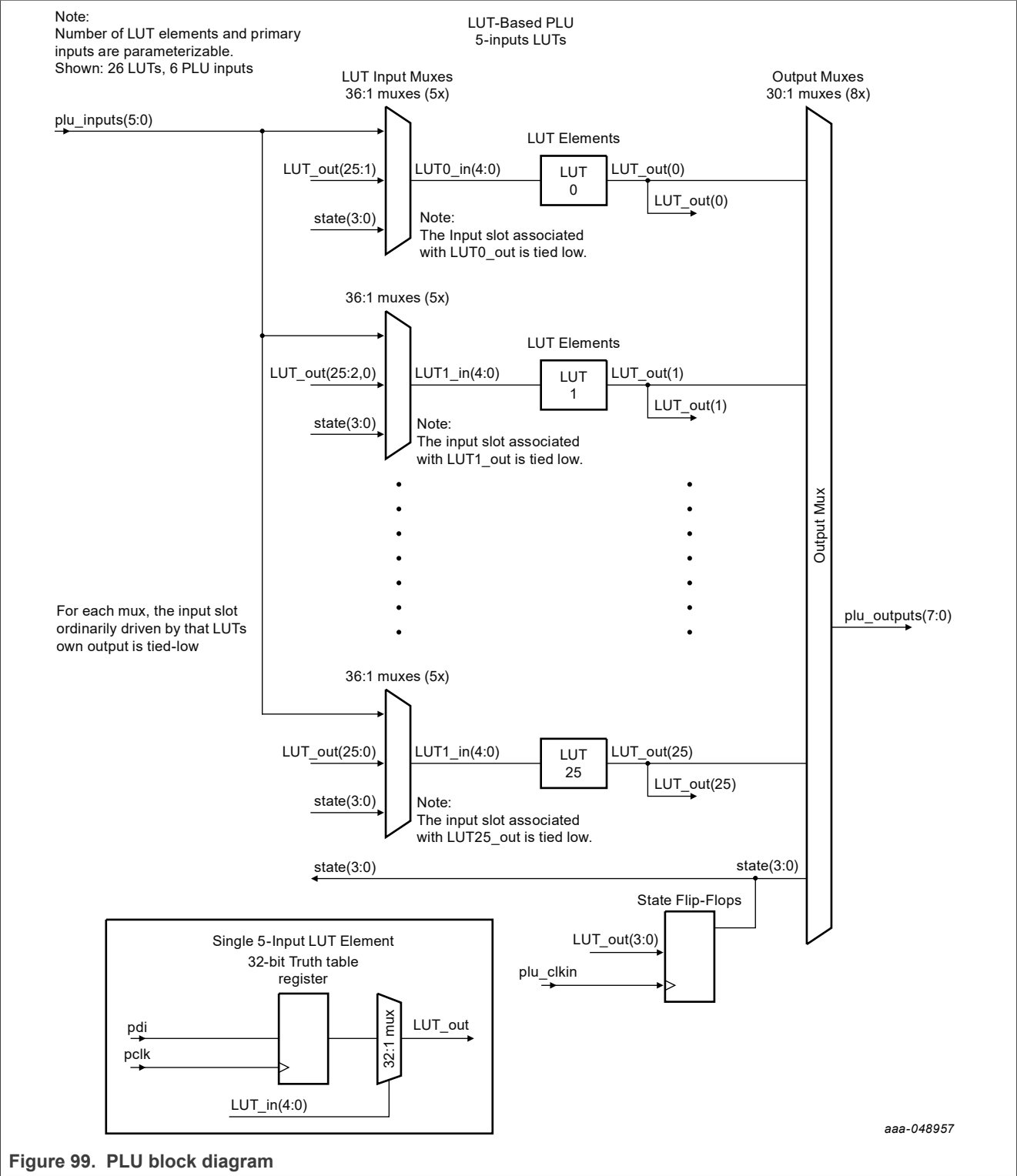


Figure 99. PLU block diagram

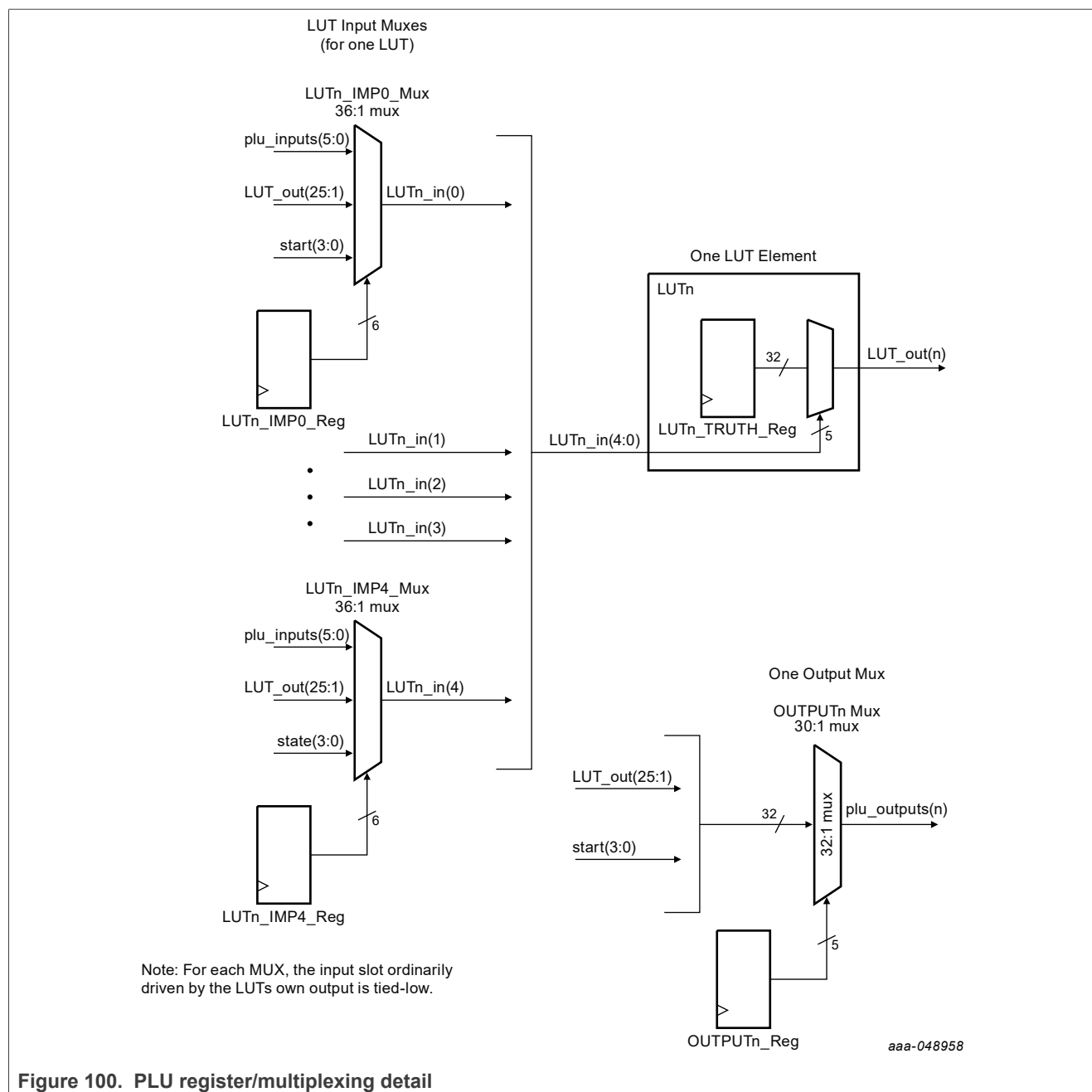


Figure 100. PLU register/multiplexing detail

38.3.1 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 that each of the LUT elements must perform. It also involves programming the input multiplexer registers to select the five inputs presented to each LUT. And it involves programming the output multiplexer register to select the eight primary outputs from the PLU module.

Programming of all of these registers is performed during initialization. Do not write to the registers during operation.

To facilitate programming of the PLU, a tool suite is provided. To implement a logic network described in a Verilog RTL input file, the tools report the values that must be written to all the registers in PLU

register descriptions (see the section on PLU register descriptions in the Register description ([Ref. 5](#)). See [Section 38.3.2](#) for a complete description of the tool flow.

Programming the I/O multiplexing through the IOCON is necessary to connect the required number of PLU primary inputs and outputs to pins (see [Section 15](#)).

38.3.2 Description of tool flow

The provided PLU programming tool suite facilitates configuring the PLU to implement the desired logic network. The input to the tool is a Verilog RTL description of the functionality to be implemented.

The output of the tool suite provides the following:

- Values to be programmed into all LUT INPUT MUX registers.
- Values to be programmed into all LUT TRUTH registers.
- Values to be programmed into all OUTPUT MUX registers.
- Error response if the described network cannot be implemented (most likely due to an excessive amount of logic or the use of more than four flip-flops).
- AC Timing parameters for the implemented design.

39 CRC engine (CRC)

The CRC engine is available on MCX W23 devices.

39.1 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 one's complement programmable setting for input data and CRC sum
- Programmable seed number setting
- Supports CPU PIO back-to-back transfer
- Accepts variable size data width per write: 8, 16 or 32-bit
 - 8-bit write: 1-cycle operation
 - 16-bit write: 2-cycle operation (8-bit x 2-cycle)
 - 32-bit write: 4-cycle operation (8-bit x 4-cycle)

39.2 Basic configuration

Set the CRC bit in the AHBCLKCTRL0 register. See the section "AHB clock control 0 (AHBCLKCTRL0)" in the Register description ([Ref. 5](#)) on how to enable the clock for the CRC engine.

39.3 Pin description

The CRC engine has no configurable pins.

39.4 General description

The CRC generator (with programmable polynomial settings) supports several commonly used CRC standards.

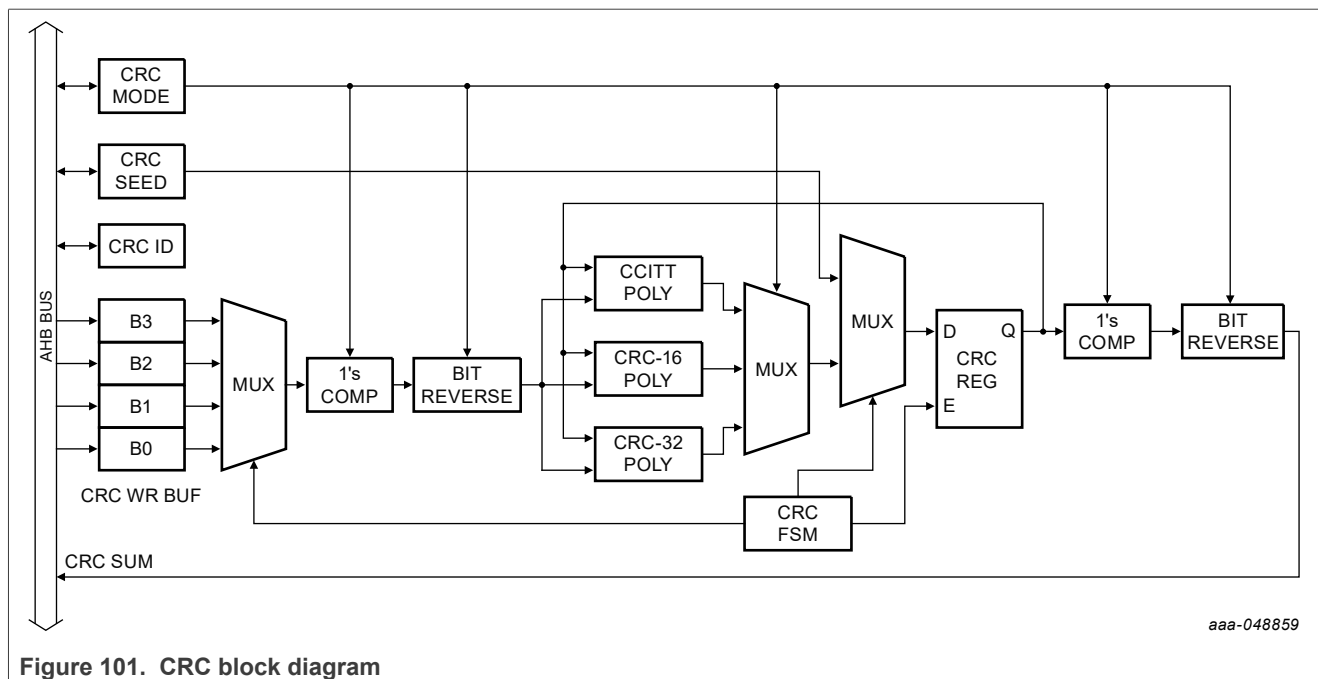


Figure 101. CRC block diagram

39.5 Functional description

The following sections describe the register settings for each supported CRC standard.

39.5.1 CRC-CCITT setup

- Polynomial = $x^{16} + x^{12} + x^5 + 1$
- Seed value = 0xFFFF
- Bit order reverse for data input: NO
- One's complement for data input: NO
- Bit order reverse for CRC sum: NO
- One's complement for CRC sum: NO
- CRC_MODE = 0x0000 0000
- CRC_SEED = 0x0000 FFFF

39.5.2 CRC-16 setup

- Polynomial = $x^{16} + x^{15} + x^2 + 1$
- Seed value = 0x0000
- Bit order reverse for data input: YES
- One's complement for data input: NO
- Bit order reverse for CRC sum: YES
- Bit order reverse for CRC sum: YES
- One's complement for CRC sum: NO
- CRC_MODE = 0x0000 0015
- CRC_SEED = 0x0000 0000

39.5.3 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
- Bit order reverse for data input: YES
- Bit order reverse for data input: YES
- One's complement for data input: NO
- Bit order reverse for CRC sum: YES
- One's complement for CRC sum: YES
- CRC_MODE = 0x0000 0036
- CRC_SEED = 0xFFFF FFFF

40 Trusted execution environment (TEE)

TrustZone for Armv8-M and trusted execution environment are available on all MCX W23 devices. This section discusses basics of TrustZone for Armv8-M and then explains in detail the additional features of MCX W23 that extend TrustZone security foundation to the system, yielding a complete trusted execution environment.

40.1 Features

- CPU0 is Arm Cortex-M33 with TrustZone support enabled
- Attribution units (SAU, IDAU)
- Secure MPU, secure NVIC, secure SYSTICK, secure stack pointer (banked)
- Secure memory map aliasing
- Support for Arm AMBA 5.0 AHB secure bus
- Secure AHB bus controller
- Memory and peripheral protection checkers (MPC, PPC)
- Security attribution wrapper for AHB masters (MSW)
- Secure DMA and DMA masking
- Secure GPIO and GPIO masking
- Secure debug

Note: *The AHB secure controller must be configured and its security check must be enabled for proper TrustZone functionality.*

40.2 Functional description

40.2.1 TrustZone for Armv8-M

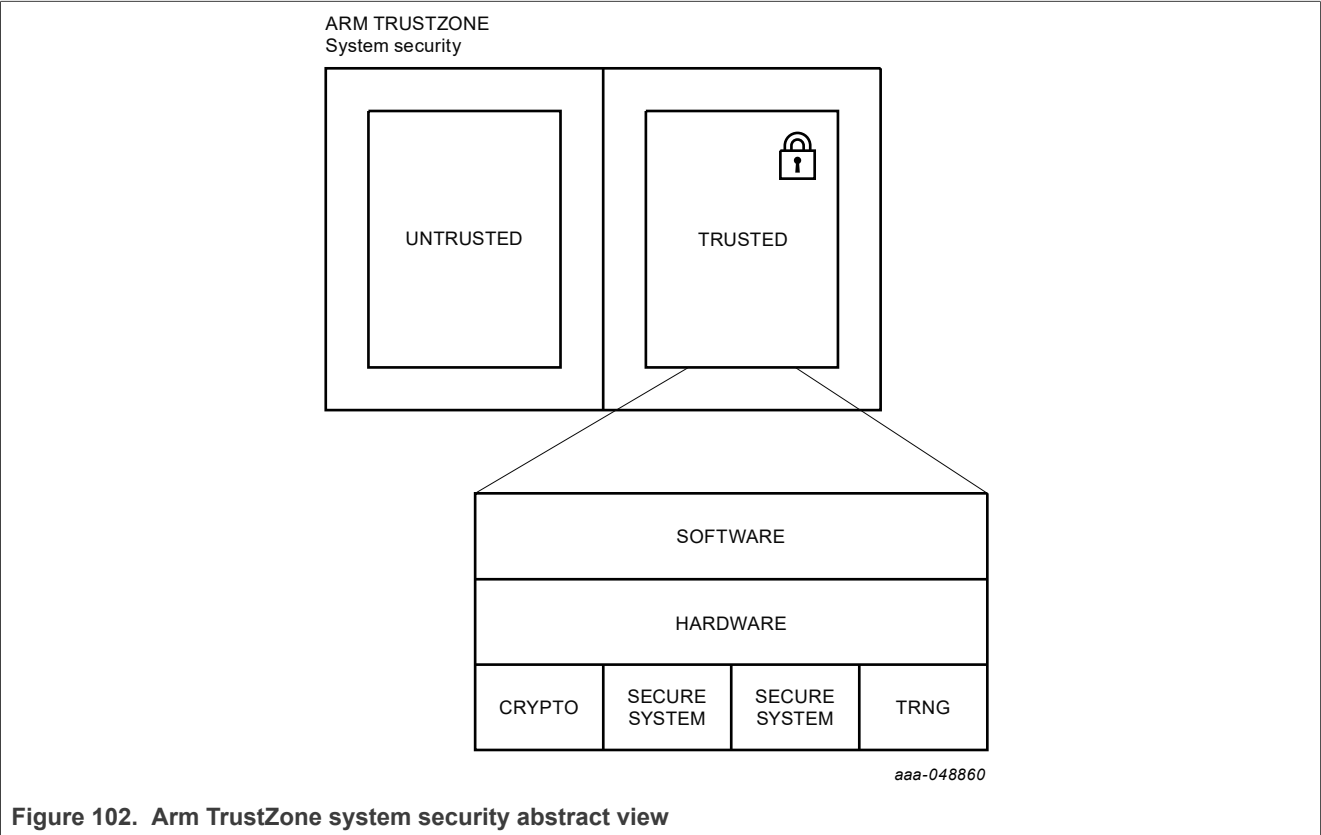


Figure 102. Arm TrustZone system security abstract view

TrustZone for Armv8-M is a feature on the Arm Cortex series that enables the execution of trusted (secure) software. It also enables access control isolation of trusted resources in a separate environment, distinct from nontrusted (nonsecure) software and resources, while running on the same CPU. It is achieved by segmentation of memory arrays and peripherals into either secure (S) or nonsecure (NS). TrustZone for Armv8-M is optimized for energy-efficient embedded applications that require real-time responsiveness.

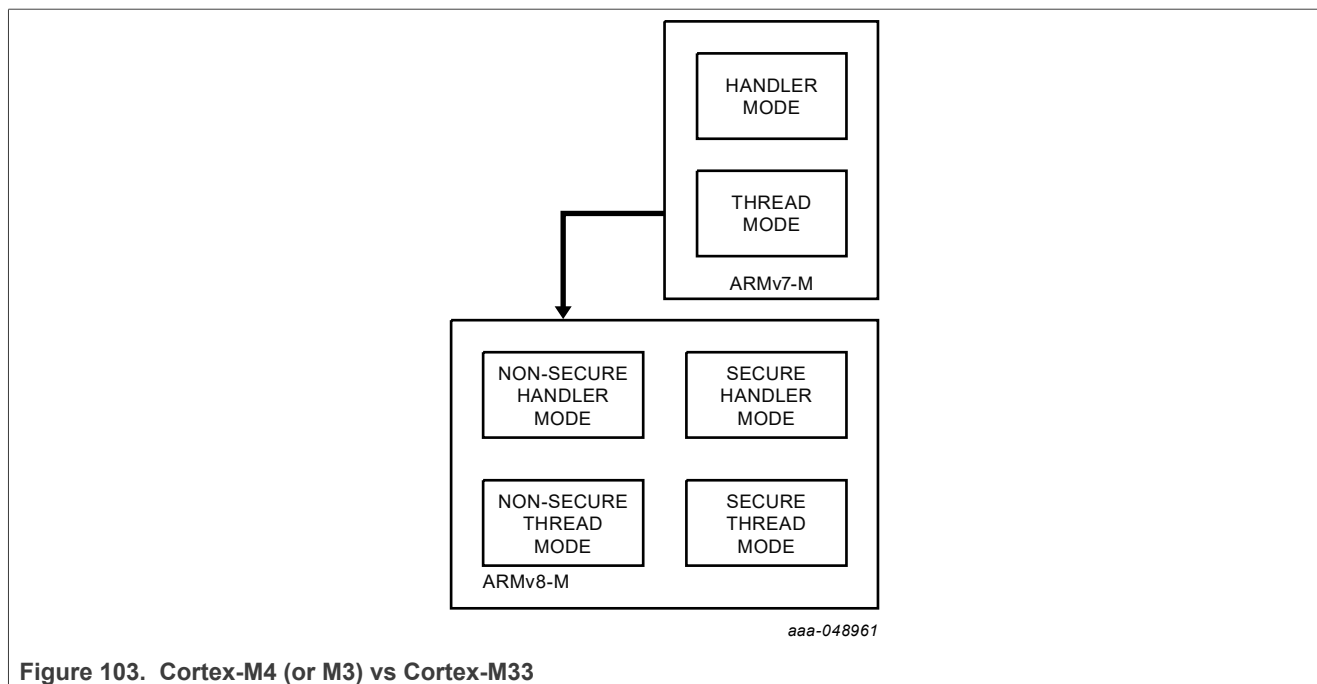
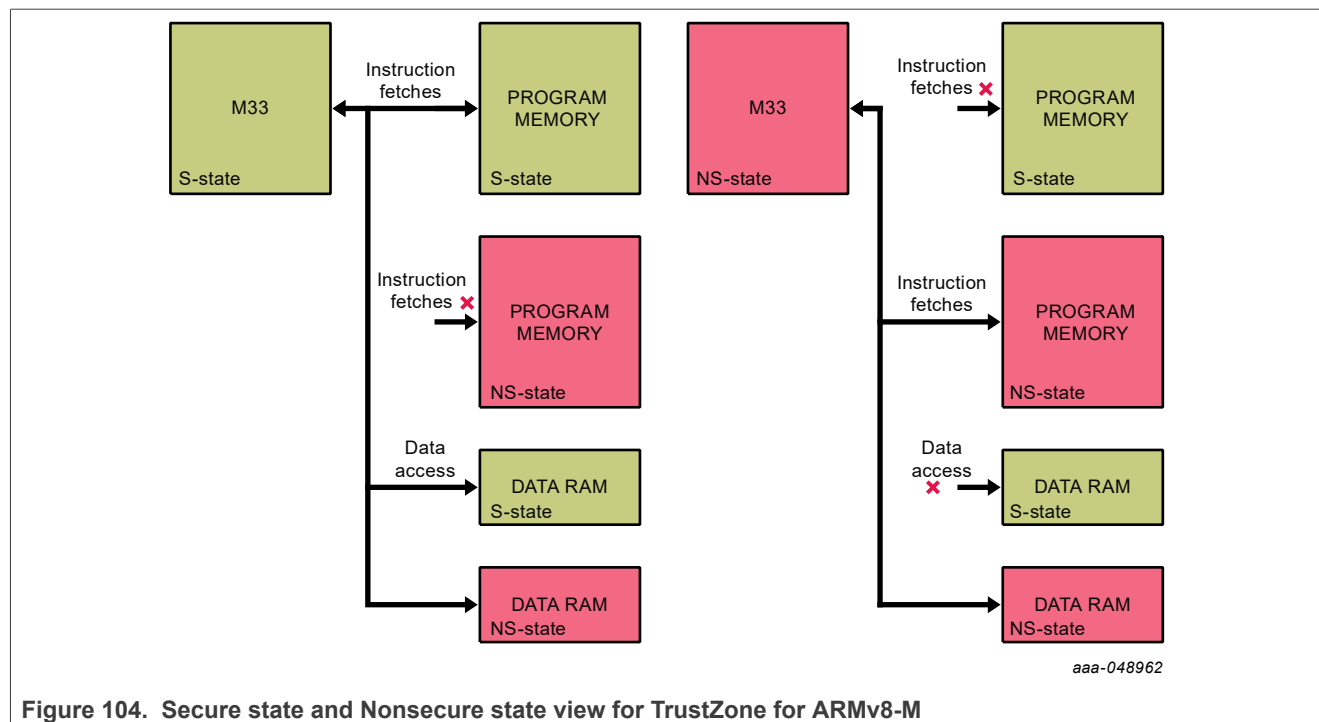


Figure 103. Cortex-M4 (or M3) vs Cortex-M33

More details on TrustZone for Armv8-M can be found in "TrustZone[®] technology for Arm[®] v8-M architecture version 1.0" document at arm.com.

If TZ-M functionality is enabled, the following rules apply to the CM33 core:

- CM33 CPU in secure state (CPU-S) can execute instructions from secure memory (S-memory) only. It cannot execute from Nonsecure 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-memory and NS-memory, as well as execute data writes to S-memory 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 executes data writes to NS-memory only. CPU-NS cannot access data from S-memory.



In summary:

- NS application code trusts that S application code does not corrupt or modify NS code or data, either inadvertently or on purpose, to create a malfunction or a hazard.
- S application code does not trust NS application code. So, it does not execute NS application code. S application code also does not trust NS data and is careful with it. Interrupt handlers can modify it. Access through NS code to secure code and data is also prevented.

To support secure state, the Cortex-M33 architecture has been extended to include secure MPU, secure NVIC, secure systick, and secure stack pointer with stack-threshold check.

40.2.1.1 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 nonsecure: Branches to an address in NS memory.
- BXLNS: Branch with link exchange nonsecure: Calls a subroutine in NS memory.

On executing the BXNS or BXLNS instruction, the CPU automatically switches its state from secure to nonsecure (CPU-NS) and so be in the correct state for executing code from NS memory.

When the CPU is in NS state, it cannot access S-memory directly. However, TZ-M provides a gateway into S-memory for NS application code using a special region called nonsecure callable (NSC). NSC region lies in S-memory. So, to execute instructions in this region, the CPU must be in CPU-S state. The NSC region of S-memory provides a gateway for NS application code to access functions in S-memory.

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 the secure function that is callable from the nonsecure code. When calling into the NSC region, the CPU-NS must access an address with the SG instruction. The SG instruction is the only instruction in the NSC region that can be executed from the CPU-NS. On executing the SG instruction, the CPU state switches from CPU-NS to CPU-S. Then, the CPU-S executes

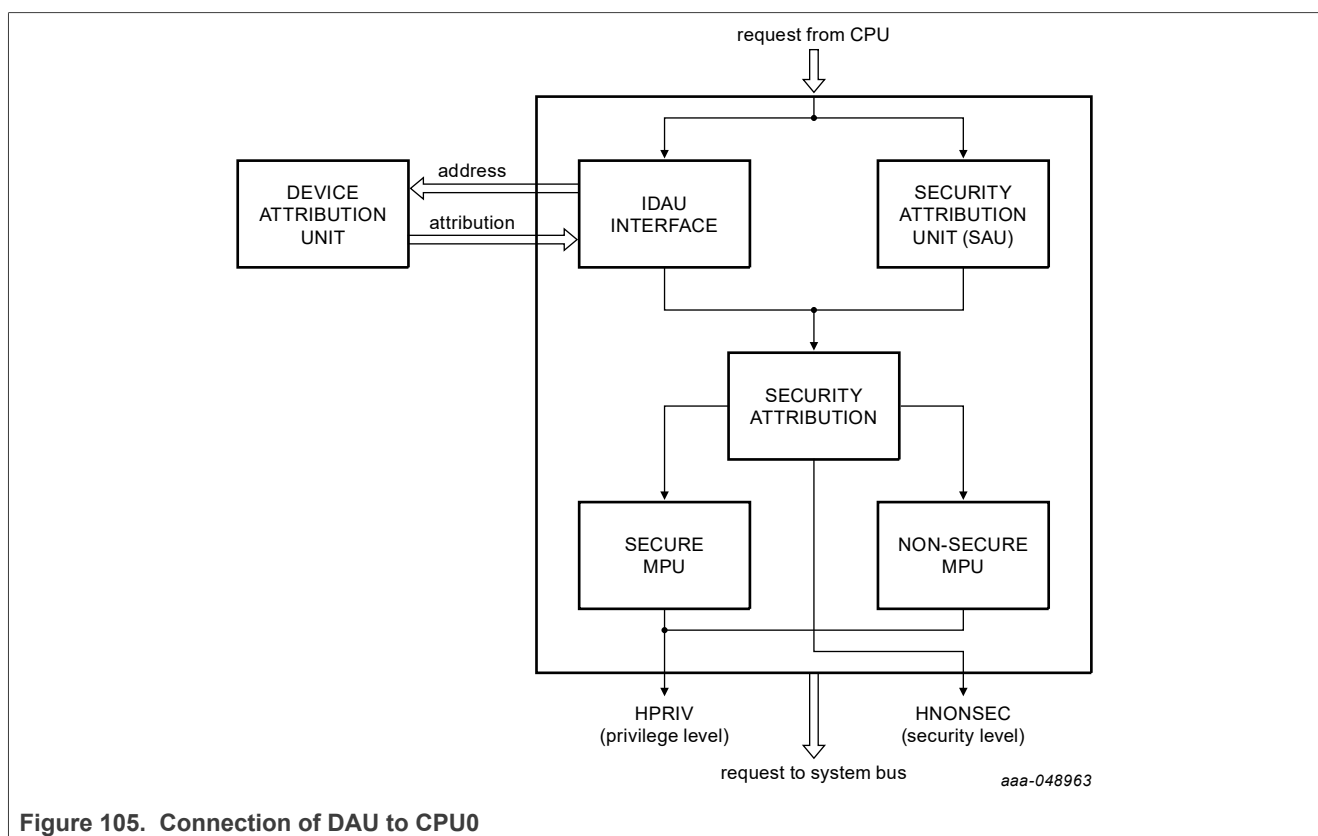
the veneered call to a secure function in S-memory. If the CPU-NS calls into an address in the NSC region that is not an SG instruction, an exception fault is created and the CPU enters the secure state.

The secure application code developer creates function calls to S application code inside the NSC region, allowing the NS application to use functions inside the S-memory.

40.2.2 Attribution units

TrustZone for Armv8-M implementation consists of the security attribution unit (SAU) and implementation defined attribution unit (IDAU). The device attribution unit (DAU) connects to CPU0 via an IDAU interface, as shown in [Figure 105](#).

The combination of SAU and IDAU assigns a specific security attribute (S, NS, or NSC) to a specific address that CPU0 requests. This attribute, together with the CPU security state, defines the access permission for CPU0. The secure AHB controller then compares that permission to the security level that a specific checker for the memory range or that specific peripheral that is being accessed requires. The secure AHB controller is discussed in [Section 40.2.3.4](#).



40.2.2.1 Device attribution unit (IDAU)

The MCX W23 implements a simple device attribution unit that divides the whole memory map into secure and non-secure regions. It is external to the CPU. All peripherals and memories are aliased at two locations.

- Address 0x0000_0000 to 0x1FFF_FFFF
 - Non-secure (always) - makes it possible to create an NSC region
- Address 0x2000_0000 to 0xFFFF_FFFF
 - Non-secure if HADDR[28] = 0

– Secure if HADDR[28] = 1

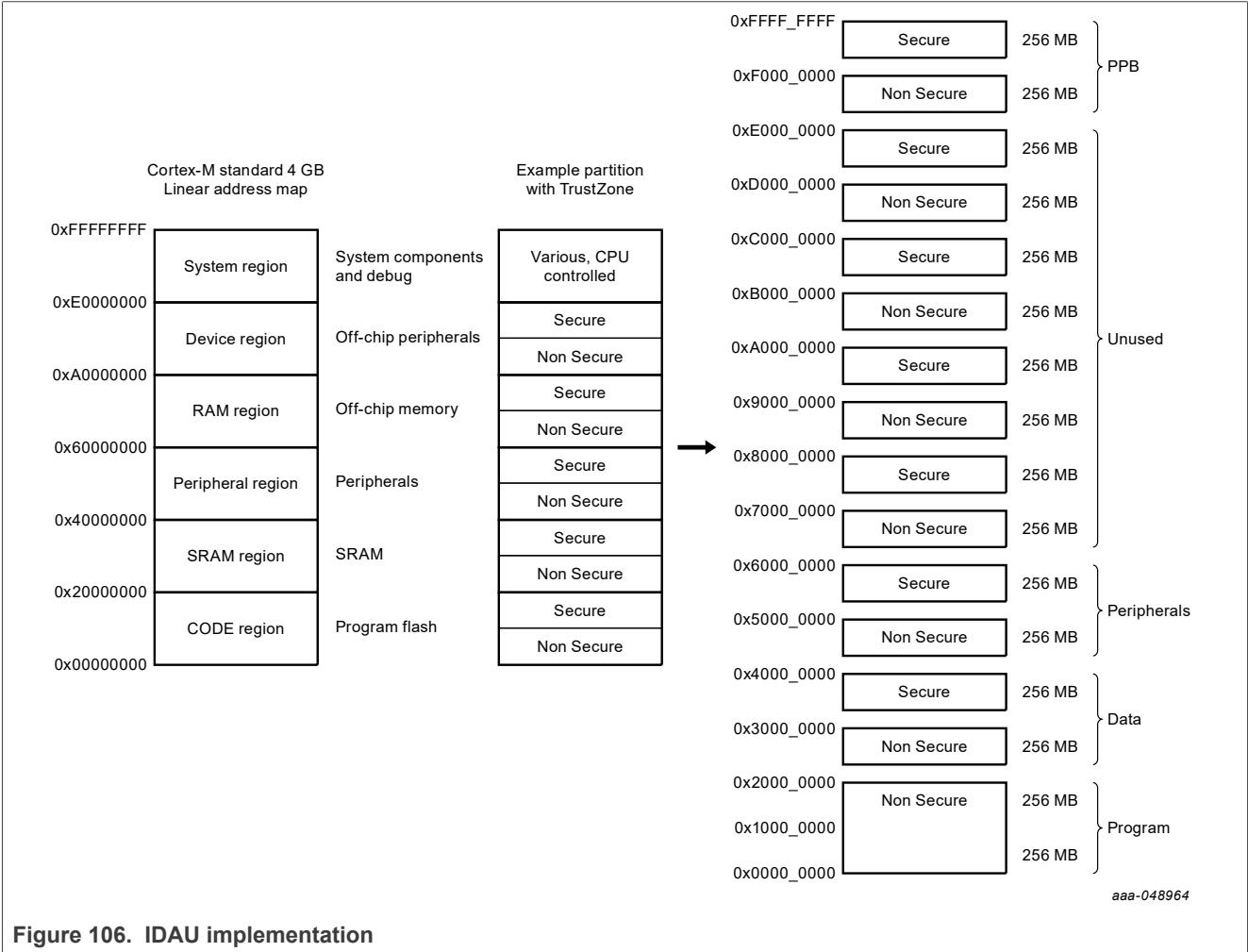


Figure 106. IDAU implementation

40.2.2.2 Security attribution unit (SAU)

The SAU is internal to CPU0 (CM33 with TZ). It monitors all accesses from CPU0 to addresses in the memory map. It does not monitor accesses from bus masters other than CPU0.

The SAU supports up to eight regional descriptors. Each descriptor allows setting the security state for a specific memory region from the following attributes.

- S: Secure.
- NS: Nonsecure.
- NSC: Nonsecure callable.

However, the range extending from 0xF000_0000 to 0xFFFF_FFFF is fixed because secure and SAU cannot program it to be NSC.

CPU0 can only configure the SAU when it is in the secure state. When enabled, the SAU marks all addresses as S by default. Only secure application code can program the descriptor to create NSC or NS regions.

The IDAU works with the SAU to assign a specific security attribute (S, NS, or NSC) to a specific address. The IDAU and SAU respond to a specific address. The CPU0 selects the higher of the two security attributes, where

the highest state is secure and the lowest state is NS. The SAU defines the NSC attribute. In IDAU, NSC area can be defined as NS. Regions are aligned to 32-byte boundaries.

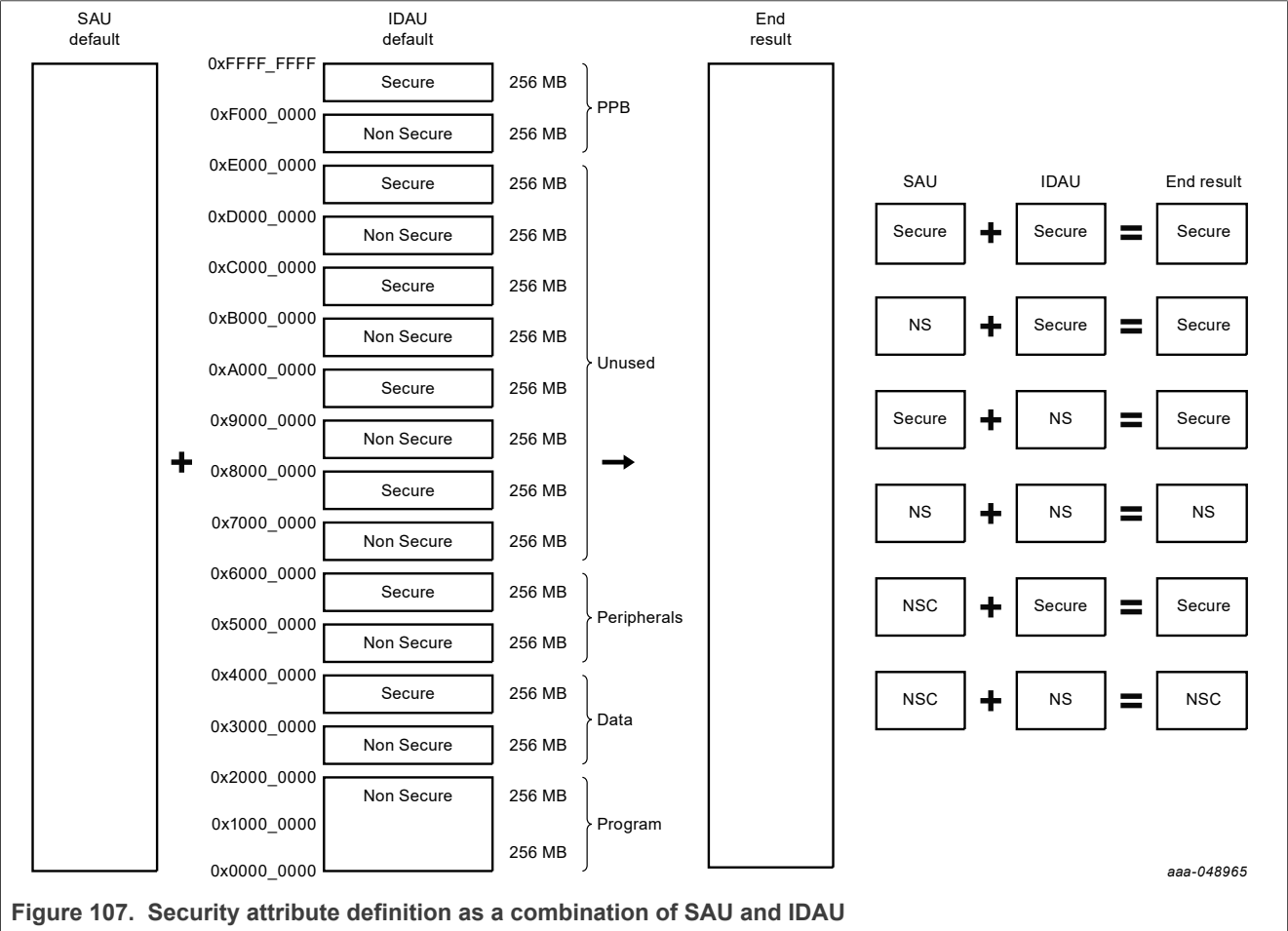


Figure 107. Security attribute definition as a combination of SAU and IDAU

From a memory-map perspective, the NS address space is an alias of the secure address space for the same physical program memory.

For instance, for the 64 kB address space at the physical addresses 0x0000_0000 to 0x0000_FFFF (that is, in the code region):

- If that range is configured as NS, only nonsecure application code can fetch instructions there, using the nonsecure alias addresses 0x0000_0000 to 0x0000_FFFF (where bit 28 of the address is 0).
 - If that range is configured as S, only secure application code can fetch instructions there, using the secure alias addresses 0x1000_0000 to 0x1000_FFFF.
- Similarly, secure application code accesses all (S and NS) peripherals in the 0x5000_0000 to 0x5FFF_FFFF space (address bit 28 = 1). NS application code accesses NS peripherals at 0x4000_0000 to 0x4FFF_FFFF space. Details of SAU programmable registers can be found in the Arm CM33 documents.

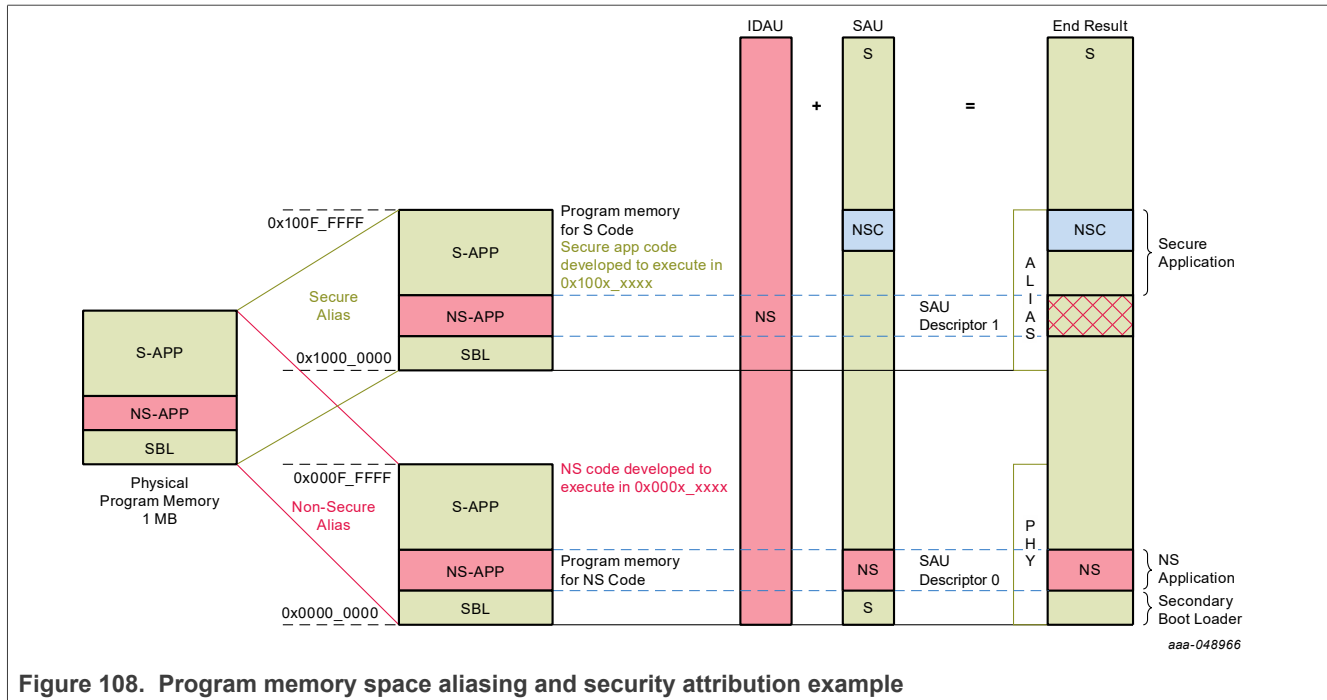


Figure 108. Program memory space aliasing and security attribution example

40.2.2.3 Region number and test target (TT) instruction

The IDAU and/or the SAU generate a region number (RN) for each region that they define. Application code can use the RN to determine if a memory access spans different regions, possibly with different security attributes. RN is an 8-bit number. In MCX W23, IDAU returns the region number as:

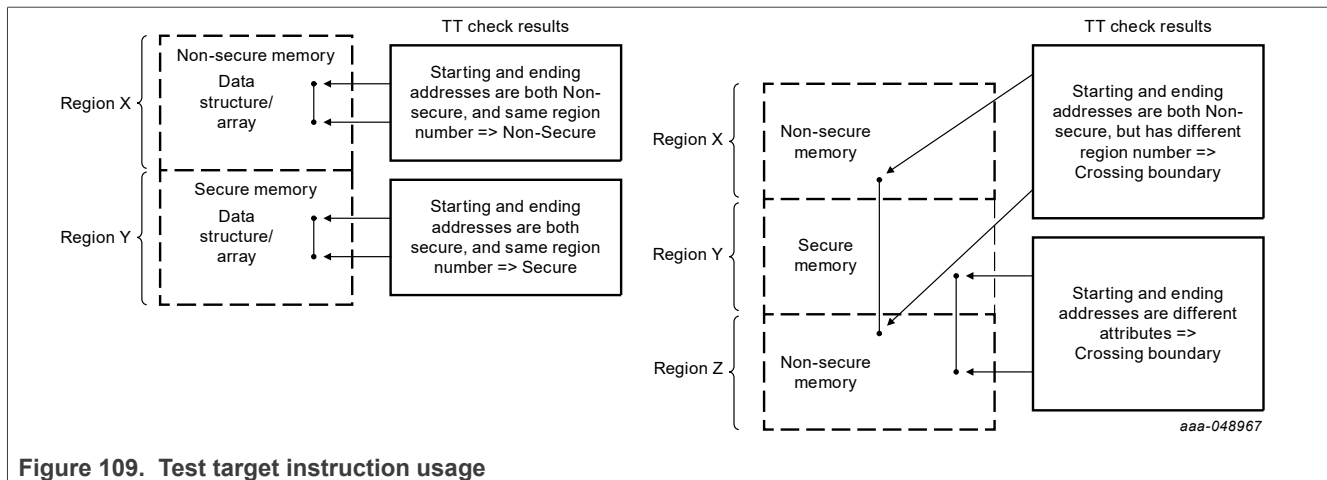
```
IDAU_RN[7:0] = ({4'h0, idau_addr_a[31:28]})
```

The TT instruction can be used to reveal, for a specific address:

- Its security level (S or NS)
- Its privilege level (privileged or unprivileged)
- Its region number (RN)

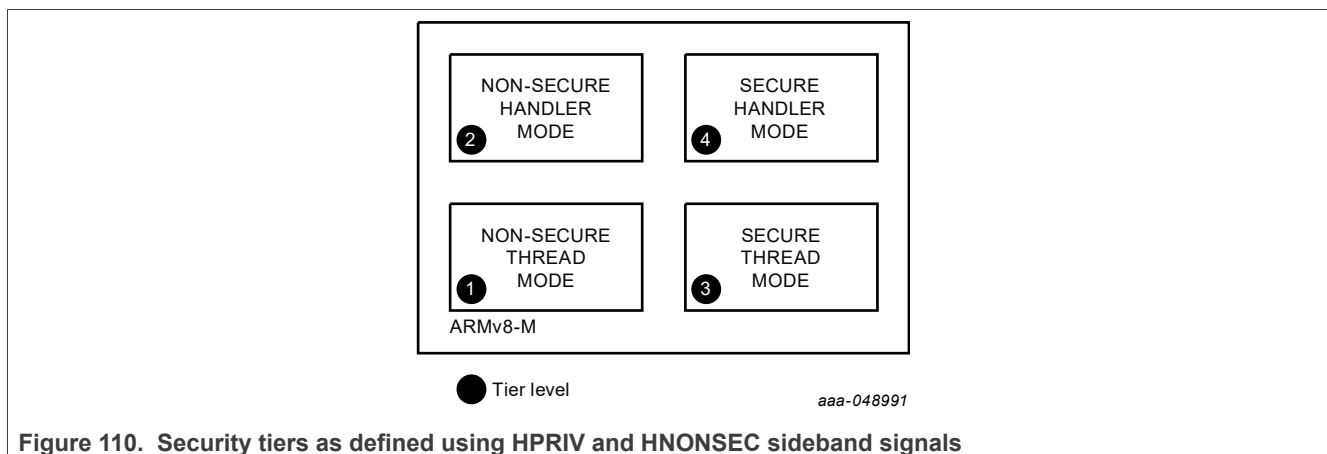
To determine if the whole contiguous memory address range has the same security attributes, the application can use the test target (TT) instruction on the start address and end address of a region.

A use case for this mechanism ensures that an address range that NS code computes lies entirely in the NS region, when secure code has to access that part of non-secure memory. It helps to prevent that secure code handles a data array that is not entirely located in non-secure space.



40.2.3 Secure AHB bus and secure AHB controller

The Arm CM33 TZ-M implementation consists of the IDAU and SAU modules, which filter address access from the CPU based on the specific security attribute (S, NS, or NSC) assigned to that address. To support secure trusted execution at system level, the MCX W23 implements a second layer of protection with a secure AHB bus controller.



The secure AHB controller provides access policies for all bus slaves via checker functions (MPC and PPC). All masters on MCX W23 output security sideband signals HPRIV (privileged) and HNONSEC (secure access) as an indication of security attributes for access to a given address. The secure AHB bus processes these signals and compares them against security attributes set for slaves in the secure AHB controller. Access is granted if, for a requested data or instruction access, the security attribute at the side of the master does not violate the security attribute of the slave that is accessed. If it does violate the security attribute of the accessed slave, a security violation interrupt is raised. To handle the violation, CPU0 switches to secure mode.

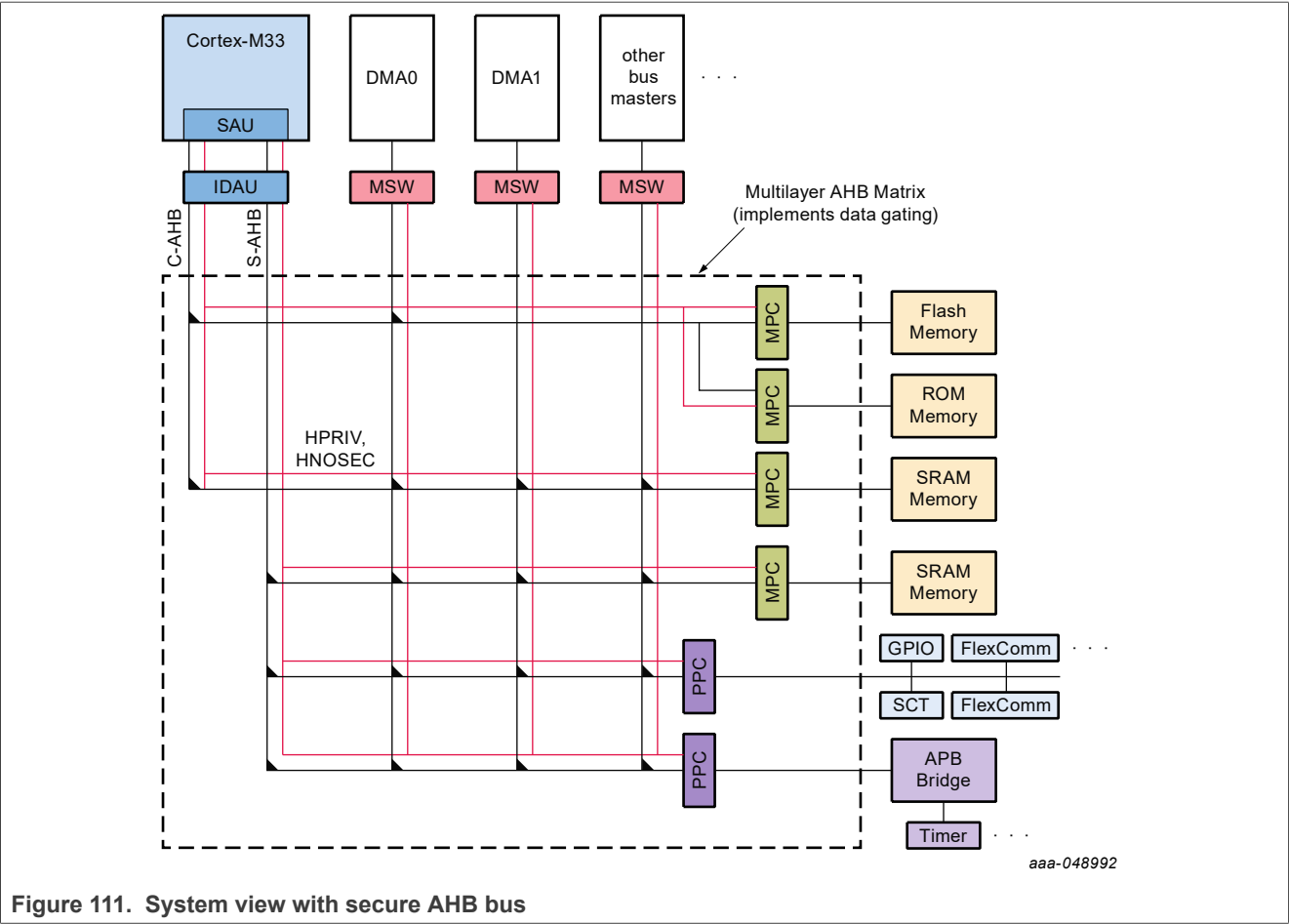


Figure 111. System view with secure AHB bus

These HPRIV and HNONSEC sideband signals create four security tiers as depicted in [Figure 111](#). Data accesses are allowed from higher tier master to same or lower tier slave. However, instruction accesses are stricter, a master can only access a slave at the same security tier. A special programmable option is available that allows treating all accesses in the system as instruction. It means that the data access checker can also be as strict.

This protection is achieved using three primary components.

40.2.3.1 Memory protection checkers (MPC)

On the MCX W23 on-chip flash, ROM and RAM can be protected against access from application code running at a lower security tier, using memory protection checkers (MPC). ROM and each RAM bank have associated MPC. Internal flash has one MPC. SPIFI external memory has one MPC as well. To offer more granularity for security tier assignment and filtering, flash, ROM, and each RAM bank are divided into smaller subregions. Each subregion can be assigned to an individual security tier by programming the corresponding registers in the secure AHB controller.

Table 205. Security tier granularity for on-chip memories

Memory	Total size	Subregion size	Subregions count
flash	1 MB	64 kB	16
flash PFR	32 kB	32 kB	1
ROM	256 kB	8 kB	32

Table 205. Security tier granularity for on-chip memories...continued

Memory	Total size	Subregion size	Subregions count
SRAM-X	16 kB	4 kB	4
SRAM-0	16 kB	4 kB	4
SRAM-1	32 kB	4 kB	8
SRAM-2	64 kB	4 kB	16
SPIFI (off-chip)	128 MB	4 MB	32

40.2.3.2 Peripheral protection checkers (PPC)

On the MCX W23, all peripherals on the AHB slave port and on the APB bridge can be protected against access from the application with lower tier security, using peripheral protection checkers (PPC). Each AHB port has an associated PPC that offers granularity at individual slave levels for security tier assignment and filtering. Each APB bridge has an associated PPC that offers granularity for security tier assignment and filtering for each APB peripheral. It can be set by programming corresponding registers in the secure AHB controller.

40.2.3.3 Master security wrapper (MSW)

When the TrustZone feature is configured, TrustZone for Armv8-M offers IDAU functionality for CM33. However, IDAU is not available for any other master on MCX W23. A special wrapper master security wrapper (MSW) is implemented for each AHB master, except for the CPU.

MSW allows an application to set a security attribute for each master. There are two categories of MSW:

- Simple master
Bus masters that can only perform data access: DMA0, DMA1, and hash-AES.
- Smart master: not applicable on this platform.

MSW for simple master enables strict checking by default. Secure data accesses can only access secure memory. A programmable option to disable strict checking allows data access from a secure master to nonsecure memory.

The security level, as defined in MSW, are supposed to be static. The application has programmed it in the secure AHB controller once and locked it until the next system reset.

Hash-AES is one exception. This master allows dynamic reprogramming of the security tier to allow secure and nonsecure code to use the functionality. To guard against malicious intent, special measures are in place within the module. Only a master with higher or the same security tier can update the hash-AES security attributes. The new attribute cannot be higher than the attribute of the programming master either. Before switching security attributes, buffers within the module are flushed. The dynamically programmed security tier is compared with the static security tier programmed in the secure AHB controller. The higher of both is the final security tier of the hash-AES engine.

If a master is programmed to be a secure master, it must output addresses where bit 28 is 1.

40.2.3.4 Secure AHB controller

The secure AHB controller is a module on MCX W23 at memory offset 0x400A C000. It allows programming security attributes for all MPCs and PPCs in addition to MSWs.

It also supports locking of the SAU setting, secure and nonsecure MPU settings (MPU_S/MPU_NS), and secure and nonsecure vector offset settings (VTOR_S/VTOR_NS) for the CPU. It enables the boot ROM to safeguard certain security features and to disable the possibility of enabling them dynamically, either unintentionally or with malicious intent.

It also supports register programming for GPIO masking and interrupt masking. Details are described in the section "AHB security control (AHB_SECURE_CTRL)" in the Register description ([Ref. 5](#)).

When a security violation is detected, the secure AHB controller module raises an interrupt. It also logs the violation information, such as the address that was accessed on an AHB slave port when the violation occurred and the access type and security attributes of the master that generated the unauthorized access.

Only an application that has secure and privileged access rights can write to secure the AHB controller to configure security attributes. So, from an application perspective, a highest-tier (tier-4) thread from CPU0 can program security attributes for system slaves.

Registers programmed in the secure AHB controller are retained during deep-sleep and power-down with CPU retention. However, after wake-up from power-down without CPU retention and deep-power down mode, these registers must be reprogrammed.

40.2.4 Interrupt, DMA, and GPIO: Secure instance and masking

The MCX W23 has one CPU with TrustZone. By default, the CPU has access to all interrupts. To safeguard a secure application, an interrupt masking feature is implemented on the MCX W23. The CPU has a TZ feature and so has secure NVIC (NVIC_NS) and nonsecure NVIC (NVIC_NS). To configure any interrupt as secure interrupt, it has internal programmability, making it only visible to NVIC_S and masking it from NVIC_NS. For more details, see the Arm CM33 documents.

The MCX W23 has two DMA instances. DMA0 offers 23 channels and DMA1 offers 10 channels. Either of the DMAs can be selected as secure DMA. This selection depends on the DMA requirements of the relevant secure peripherals selected. To prevent DMA request from secure peripherals to be visible to nonsecure DMA, a DMA masking feature is implemented. DMA masking can be programmed using registers in [Section 18](#) or [Section 22](#).

On the MCX W23, all digital I/O pins states are readable through the GPIO module, independent of which function is chosen, using the I/O multiplexer. If a secure peripheral is connected to the interface, this readability of the GPIO pins can lead to an information leak. To safeguard incoming data from secure peripherals, GPIO masking is implemented on the MCX W23. Any digital I/O that is sensitive to information leakage can be masked using the SEC_GPIO_MASK0 register at offset 0xF80 in the secure AHB controller module.

MCX W23 has a second instance of GPIO and the PINT module on Port0 (0-22). Unlike normal GPIO, this GPIO functionality is implemented as IOMUX function and only available if selected using IOCON programming. It can be used as secure GPIO to generate certain input pattern from an external device for secure signaling. More details can be found in [Section 17](#) and [Section 20](#).

41 Security features

The secure hash algorithm (SHA), the random number generator (RNG), AES encryption/decryption registers, the PRINCE real-time encryption/decryption engine, the physical unclonable function (PUF), the universal unique identifier (UUID), and security APIs are available on all MCX W23 devices.

41.1 Introduction

To implement the security features of the device, the security system on MCX W23 has a set of hardware blocks and ROM code. The hardware consists of an AES engine, an SHA engine, a random number generator, a PRINCE engine, and a key storage block that derives keys from an SRAM-based physically unclonable function (PUF). [Figure 112](#) shows an overview of the MCX W23 security system. To encrypt or decrypt data and for hashing, the processor or the DMA engine can access all components of the system. The ROM is responsible for secure boot, in addition to supporting various security functions.

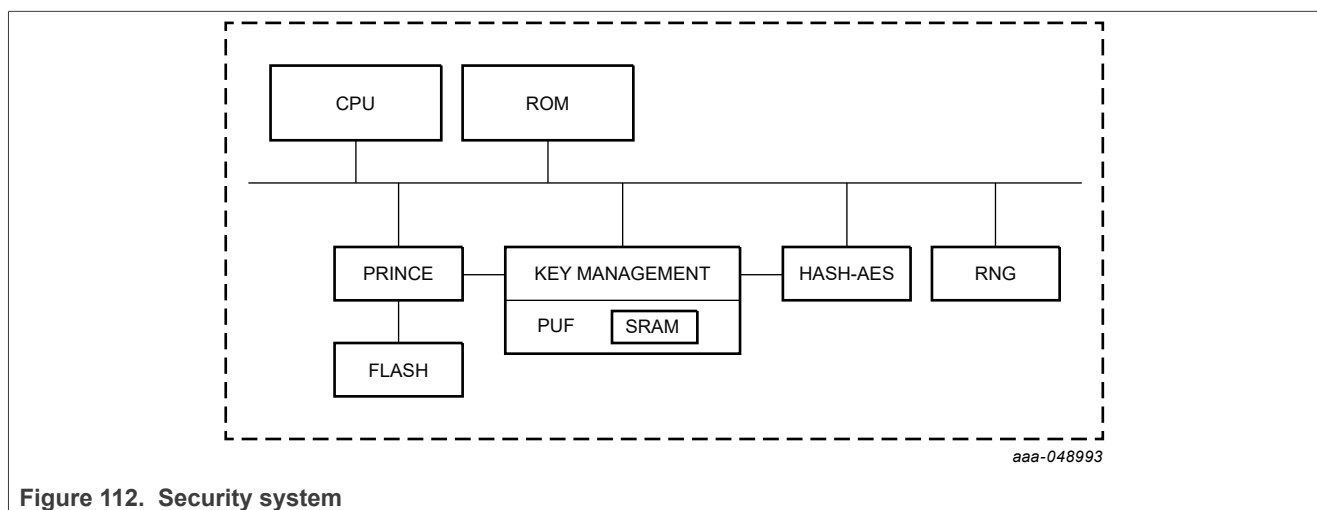


Figure 112. Security system

41.1.1 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 MCX W23 offers SRAM PUF where the PUF provides a unique key per device. It exists in that device based on the unique characteristics of PUF SRAM. [Figure 113](#) shows the block diagram of how the AES engine uses keys. PUF keys have a dedicated path to the AES engine and PRINCE engine where only the intended engine can use its key. No other mechanism by which keys can be observed is available. KEY0 feeds into AES. Keys 1/2/3/4, from the key management block, feed into PRINCE.

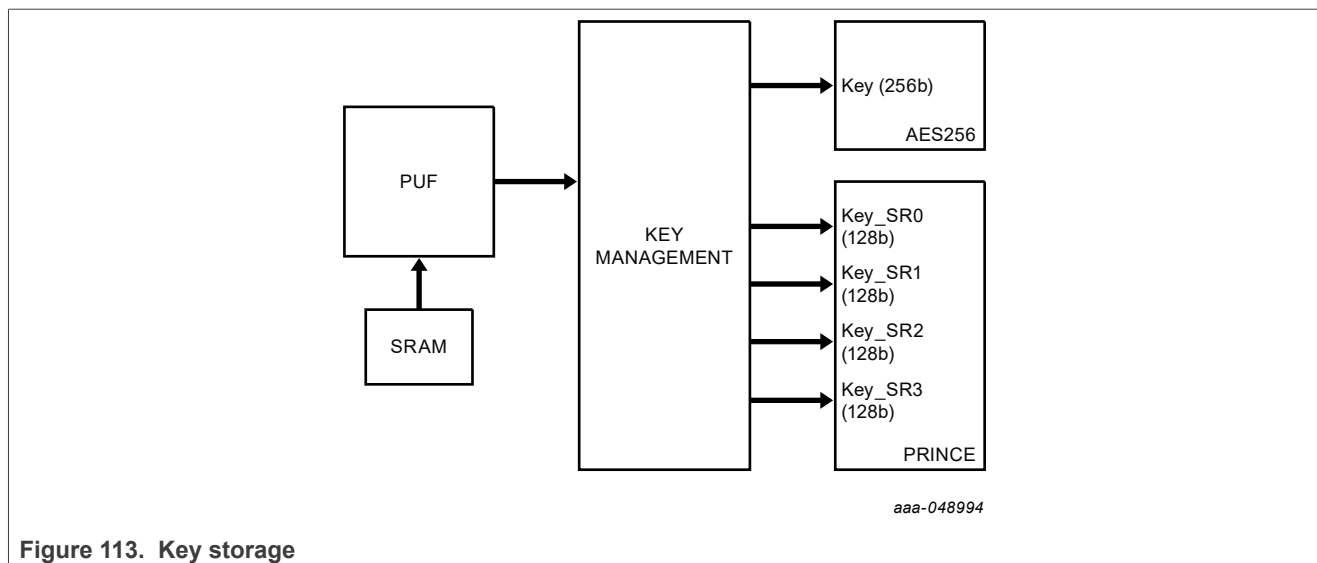


Figure 113. Key storage

41.1.1.1 PUF keys

The PUF controller provides secure key storage without storing the key. It is done by using the digital fingerprint of a device derived from SRAM. Instead of storing the key, a key code is generated, which in combination with the digital fingerprint is used to reconstruct keys that are routed to the AES engine or that software uses. The PUF controller provides generation and secure storage for keys.

41.1.2 AES engine

To protect the image content and to accelerate processing for data encryption or decryption, data integrity, and proof of origin, the MCX W23 devices provide an on-chip hardware AES encryption and decryption engine. The AES engine can encrypt and decrypt data using a key from the PUF or a key that the software provides (See [Section 41.3](#)).

41.1.3 SHA

All MCX W23 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 into 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 causes a major change in the digest output. So, for a given input message or image only one digest is available.
- No predictable way is available to modify one input to result in a specific digest. A message cannot be added, inserted, or modified to result in the same hash in any direct way.

These two properties make it useful for verifying that a message is valid or corrupted, either intentionally or unintentionally.

Hashing is used for four primary purposes:

- Core of a digital signature model, including certificates, for example, and for secure update.
- Support a challenge/response or to validate a message when used with a hash-based message authentication code (HMAC).
- In a secure boot model, which verifies code integrity.

- Verify that external memory has not been compromised.

See [Section 41.4](#).

41.1.4 Digital signatures

A digital signature combines the use of public/private keys, such as RSA or ECC, with SHA hashing. A signature is formed in the following way:

- The message or image is hashed using SHA1, SHA2, or any other hashing algorithm.
- The digest is formed into a buffer along with a header and padding. The header indicates which hashing algorithm was used (for example, SHA1). The padding fits the buffer to the size of the public key, for example, 256 bits, 1024 bits, and 2048 bits.
- The buffer is signed (encrypted) using the private key.

Note: That signing is a reverse of the normal public or private key where anyone can encrypt using the public key, so only the private key holder can decrypt. In this case, the model is reversed, so that only a private key holder can sign and anyone can verify if it is from the private key holder.

To verify the signature, the following steps are used:

1. The message or image is hashed using the same hashing algorithm as the one that was initially used for creating the signature.
2. The signature is decrypted using the public key.
3. 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. So, the signature can be stored in an external flash file along with the image. It can then be verified using a stored copy of the public key.

41.1.5 Hash-based message authentication code (HMAC)

An HMAC can be achieved on MCX W23 using a preshared symmetric key and hashing. It is a way of verifying the integrity and authenticity of a message (encrypted or not). It can also be used for challenge or response. Both sides must have a preshared key that is just a shared secret value and not an encryption key.

The HMAC is formed using three steps:

1. Hashing the message or image.
2. Taking the resultant digest and combine with the (hashed) key and some padding.
3. Hashing the combination of digest, key, and padding. The resulting 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 so can trust the data that was hashed.

HMACs are faster than signatures. However, they work only with preshared keys, which must not be leaked or lost (unlike a public key). The HMAC key can be shared dynamically using trust models like Diffie-Hellman or maybe a board-unique key that two devices share.

41.1.6 True random number generator (TRNG)

True random number generators (TRNG) are used for cryptographic, modeling, and simulation applications, which employ keys that must be generated in a random fashion (see [Section 41.5](#)).

41.1.7 Universally Unique Identifier (UUID)

The MCX W23 stores a 128-bit IETF RFC4122 compliant nonsequential universally unique identifier (UUID). It can be read from flash NMPA region at location 0x0010_8B70 (NS) or 0x1010_8B70 (S) onward.

41.1.8 Boot seed

MCX W23 contains 8 boot seed registers for generating random numbers.

The true random number generator generates the boot seed on each boot. It is used as a key to an HMAC computation. It shows that the LATEST secure boot process in ROM has completed successfully. Since we use a different random boot seed on each boot, if the most recent secure boot sequence has generated that value and its HMAC or if they have been replayed, can be verified easily. The randomness guarantees the freshness of this information.

The inclusion of the hash of the secondary bootloader and, optionally, the NMPA, CMPA, and CFPA, ensures the state of the device at boot time.

A real-time library can be used to respond to attestation challenges using this boot seed.

See the Arm PSA specifications at <https://developer.arm.com/architectures/architecture-security-features/platform-security>.

41.1.9 PRINCE real-time encryption/decryption

The MCX W23 devices support 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 having to store data in RAM first and then encrypt or decrypt to another space. It operates on a block size of 64 bits with a 128-bit key.

This functionality provides confidentiality and is useful for asset protection, such as securing application code, securing stored keys, and enabling secure flash update (see [Section 41.6.1](#)).

41.2 PUF controller and key management

The PUF controller provides a secure key storage without injecting or provisioning device unique PUF root key. For more details, see [Section 41.1.1.1](#).

41.2.1 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 the enrollment process and is stored in nonvolatile memory on the device.
- Generation, storage, and reconstruction of keys.
- Key sizes from 64 bits to 4096 bits.
 - The PUF controller allows storage of keys, generated externally or on-chip, of sizes 64 bits to 4096 bits.
 - The PUF controller combines keys with digital the fingerprint of the device to generate key codes. To reconstruct an original key, these key codes must be provided to the controller. They can be stored in nonvolatile memory on the device.
- Key output via dedicated hardware interface or through register interface.
 - The PUF controller allows the assignment of 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, which is only accessible to AES and PRINCE engines. Keys with nonzero index are available through the APB register interface.
- 32-bit APB interface.
- Programmable feature to block indexes from generating new key codes.

41.2.2 Basic configuration

- The PUF block can be reset using the PUF_RST bit in PRESETCTRL2 register (see the section "Peripheral reset control 2 (PRESETCTRL2)" in the Register description ([Ref. 5](#)). However, certain registers in the key management wrapper are only reset with a global system reset, not with this IP reset.
- The clock to the PUF can be controlled using the PUF bit in AHBCLKCTRL2 register. See the section "AHB clock control 2 (AHBCLKCTRL2)" in the Register description ([Ref. 5](#)).

41.2.3 PUF controller operations

The PUF controller supports the following operations:

- **Enroll:**
The controller retrieves the start-up data (SD) from the memory (SRAM), derives a digital fingerprint, generates the corresponding activation code (AC) and sends it to the NXP key store. NXP performs this step is performed once for each device, and no other customer can perform it.
- **Start:**
The AC generated during the enroll operation and the SD are used to reconstruct the digital fingerprint. It is done after every power-up and reset.
- **Generate key:**
The controller generates a 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.
- **Set key:**
The digital fingerprint generated during the enroll/start operations and that the client design (CD) provides are used to generate a key code (KC). It is also possible to make the PUF generate a random key internally (called "intrinsic key") and output the corresponding key code. Perform this operation only once for each key.
- **Get key:**
The digital fingerprint generated during the start operation and the KC generated during a set key operation are used to retrieve a stored key. Perform this operation every time a key is required.

41.2.4 SRAM PUF power control

The SRAM used for the PUF is a separate block of memory that is only the PUF controller can access. All self-test is built into the PUF logic. It is not accessible through any chip test modes. To ensure that clean startup power is available and is controlled via the PUF PWRCTRL register (address, 0x4003 B108), the PUF SRAM has a power switch. If a previously powered PUF SRAM is turned off by clearing the bit RAMON in the PWRCTRL register, up to 400 ms delay is required before the PUF SRAM can be powered again by setting bit RAMON to 1. The required delay depends on temperature conditions and is only highest at the worst corners. ROM API implements a special scheme where the first attempt is made with a shorter delay and, if that fails, a 400 ms delay is applied in the next iteration.

41.2.5 Key management

The MCX W23 key management module supports storing an AES Key (KEY0) and four PRINCE Keys (KEY1, KEY2, KEY3, KEY4). These keys are fed into their respective IPs via a dedicated hardwired interface. Software cannot read them. Since these keys are from Index-0, the software interface cannot access them. The PRINCE requires a 128-bit key. The MCX W23 device supports up to four regions. So, four separate 128-bit keys are made available via the key management block. The AES key can be 128 bits, 192 bits, or 256 bits in length.

PUF keys for AES and PRINCE, if already loaded, are retained during deep-sleep and power-down but not retained during deep-power down. The CTRL, CFG, KEYLOCK, KEYENABLE, KEYRESET, IDXBLK, IDXBLK_DP, and SHIFT_STATUS registers are preserved and not reset during power-down.

This module supports the blocking of access to a set of indexes such that they cannot be used anymore for key generation or retrieval until next reset.

41.2.5.1 Key loading procedure

To load KEYn for the AES or PRINCE to use, follow the procedure below:

1. Write the enable value, 0x2, to the KEYn field of the KEYRESET register. It clears the associated KEYn hold and KEYn field in the SHIFT_STATUS register.
2. Write the enable value, 0x2, to the KEYn field of the KEYENABLE register. Ensure that only the intended KEYn field in the KEYENABLE register is enabled. To avoid overwriting other keys, the other KEYn fields must be disabled.
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 to the key management module on the dedicated hardware interface (see [Section 41.2.6.11](#) and [Section 41.2.7.6](#)). It is assumed that PUF initialization and start have already been performed before issuing the get key command. The requested key is loaded into the KEYn hold register, which is only visible to the AES or PRINCE.
5. If necessary, write the disable value, 0x1, to the KEYn field of the KEYLOCK register to prevent any further changes to KEYn.

41.2.6 Using PUF

This section describes steps for the setup and usage of the PUF block.

41.2.6.1 Order of operations

After power-up or reset, the PUF controller starts in one of the three initialization states, depending on its previous state (see [Figure 114](#)). 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: *NXP can only perform the enroll operation internally.*

After the start operation, set key and get key operations can be performed. The set key and get key operations can be performed repeatedly. When the device is reset the Start operation must be performed again, before performing Get Key and Set Key operations.

Note: *The generate key and set key operations can only be performed when BLOCKSETKEY = 0.*

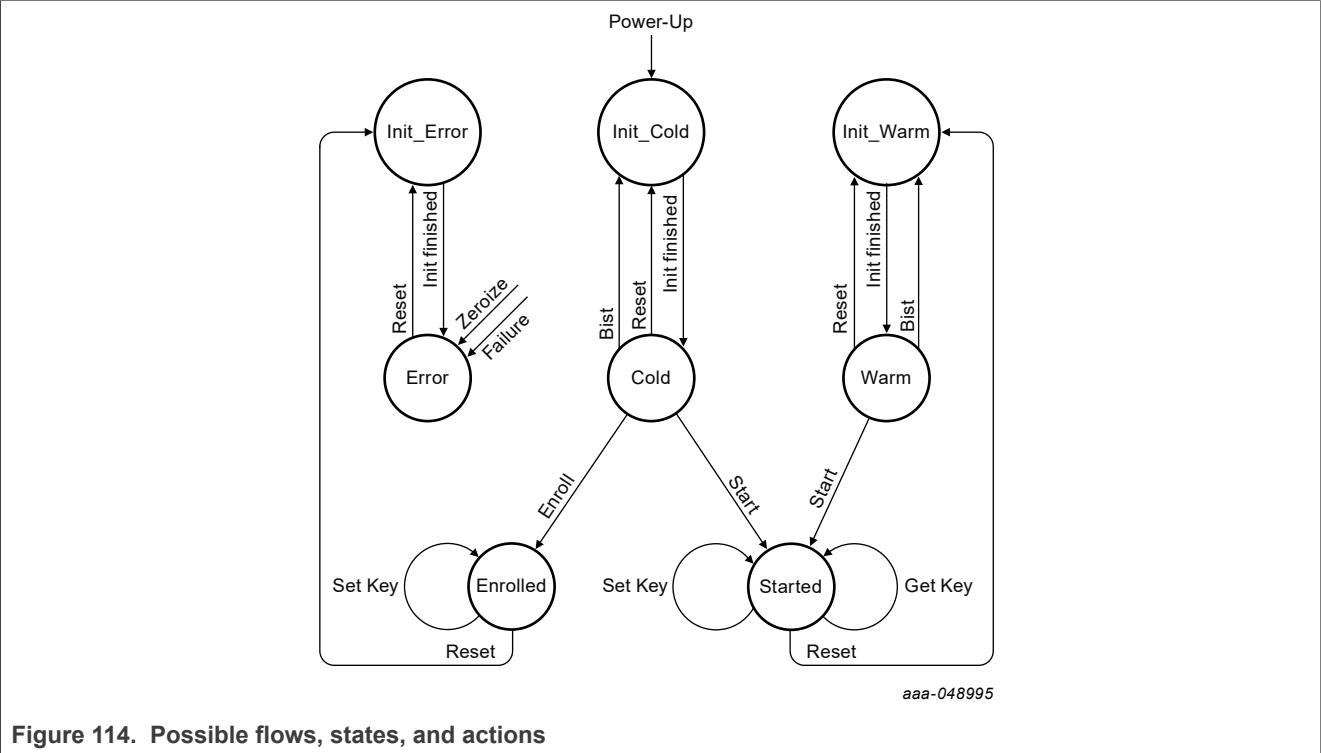


Figure 114. Possible flows, states, and actions

If a ZEROIZE operation occurs through the control register or a failure, for example, a wrong activation code the PUF controller goes to the error state. It erases all internal critical security parameters and disables communication with the PUF. The only way to leave this state is by repowering the device.

The indicative length of each operation is shown in [Table 206](#), 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 206. 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

41.2.6.2 Activation code size

The size of the activation code (AC size) is:

AC size = 9536 bits (1192 bytes)

41.2.6.3 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 bits can be stored safely.

Note: Keys generated that the PUF controller generates are randomly generated by construction. It means that generated keys cannot be used for cryptographic algorithms that require keys with a specific mathematical structure, which is typical for public key schemes like RSA. In such cases, an externally generated key must be used and stored as a user key.

The key code size (KC size in bits) depends on the key size. [Table 207](#) specifies the KEYSIZE values to use for the supported key sizes and the size of the related PUF-generated KC.

Table 207. Coding of KEYSIZE

KEYSIZE (5:0)	Value	Key size (bits)	KC size (bits)	KC size (bytes)
000001	1	64	416	52
000010	2	128	416	52
000011	3	192	416	52
000100	4	256	416	52
000101	5	320	672	84
000110	6	384	672	84
000111	7	448	672	84
001000	8	512	672	84
001001	9	576	928	116
001010	10	640	928	116
001011	11	704	928	116
001100	12	768	928	116
001101	13	832	1184	148
001110	14	896	1184	148
001111	15	960	1184	148
010000	16	1024	1184	148
...	...	...		
100000	32	2048	2208	276
...	...	...		
110000	48	3072	3232	404
...	...	...		
000000	64	4096	4256	532

41.2.6.4 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 software AES encryption module that has number 0xA assigned to it uses a key. Before the set key operation, the KEYIDX bits in the KEYINDEX register are set to 0xA. Then the set key operation is started and the key is passed to the PUF. The resulting key code (KC) is stored.

When the key is required, the get key operation is started and the KC is passed to PUF. The key index related to this KC appears on KEYOUTIDX in the KEYOUTINDEX register and the resulting key appear in the KEYOUTPUT register. Using the KEYOUTIDX value, the key can be sent to the AES target with number 0xA.

41.2.6.5 Key code header

The first 32 bits of the key code area header with information about the type of key it represents. It includes three fields (see [Figure 115](#)). The unused bits are always 0.

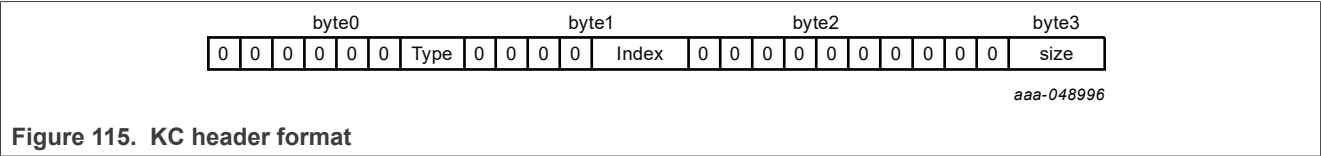


Table 208. KC header field description

Name	Description
Type	Define key type. 11: Reserved 10: Reserved 01: Generate key 00: User key
Index	Value of KEYIDX when the set key or generate key operation is in progress.
Size	Value of KEYSIZE when the set key or generate key operation is in progress.

41.2.6.6 Key byte order on the APB interface

The first word contains the first bytes and the second word the following bytes. [Figure 116](#) shows the byte 16 order within a word.

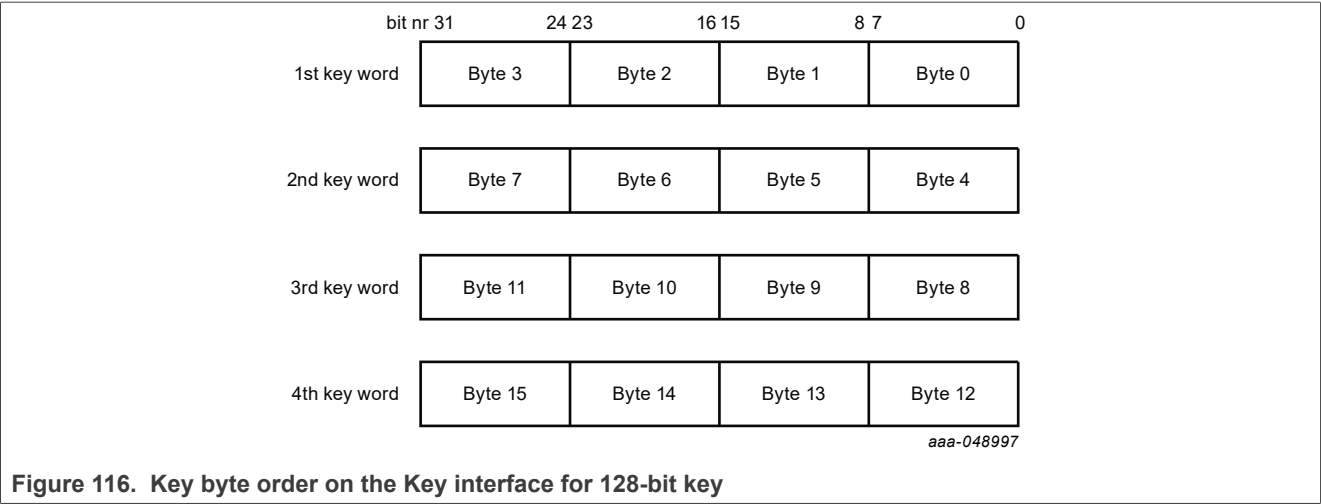


Figure 116. Key byte order on the Key interface for 128-bit key

41.2.6.7 Enroll

During the enroll operation, the SRAM start-up values are read. Based on these values, a device-specific digital fingerprint is derived and the related activation code (AC) is generated. The following steps are performed:

- 1. Software gives the enroll command to PUF by setting ENROLL bit.
- 2. PUF reads the startup values, computes a digital fingerprint and makes the related AC available through CODEOUTPUT.
- 3. After the operation is finished, the PUF deasserts BUSY and asserts SUCCESS, signaling that the enroll operation has completed successfully.

41.2.6.8 Start

During the start operation, the SRAM start-up values and the activation code are read. Based on these values, 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 start-up values and requests for the AC through CODEINPUT.
- 3. After the operation is finished, the PUF deasserts 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 enters the error state (see [Section 41.2.6.1](#) and [Section 41.2.6.13](#)).

41.2.6.9 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 the generated key, 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 deasserts BUSY and asserts SUCCESS, signaling that the generate key operation has completed successfully.

41.2.6.10 Set key

During this operation, the key size and key index are defined first and the user key is read. Based on the reading of the user key, the device-specific key code (KC) is generated. The following steps are performed:

1. Set KEYIDX and KEYSIZE to their required values.
2. Enter the user key in the KEYINPUT.
3. Give the set user key command to the PUF controller using SETKEY.
4. The KC is available through CODEOUTPUT.
5. After the operation is finished, the PUF controller deasserts BUSY and asserts SUCCESS, signaling that the set key operation has completed successfully.

Note: Keys with key index 0 are sent to the AES and PRINCE key interface. When the user key index is 0, write the user key to the KEYINPUT register with the least significant word first. User Keys with other indexes must be written to the KEYINPUT register with the most significant word first.

41.2.6.11 Get key

During the get key operation, the key code is read. The key code includes the key index and key size. The values for KEYIDX and 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 [Section 41.2.6.11](#). For more information on using KEYOUTIDX bits, see [Section 41.2.6.4](#).
3. After the operation is finished, BUSY is deasserted 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 enters the error state (see [Section 41.2.6.1](#) and [Section 41.2.6.13](#)).

Note: All key bits produced as defined in the KC must be consumed. If fewer key bits as defined in the KC are consumed, the PUF controller stays busy until the remaining bits are consumed.

Table 209. 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.

41.2.6.12 Zeroize

When the ZEROIZE bit is programmed to 1, all internal-critical security parameters are erased and the PUF controller enters the error state (see [Section 41.2.6.1](#)). No new operations can be performed until the device is repowered.

41.2.6.13 Error response

When an error other than an APB error is detected, all internal critical security parameters are erased and the PUF controller enters 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 deasserted, independent of the state of the command signals.

41.2.6.14 Key index blocking

When the index blocking register is programmed, key output from the blocked index is not possible. However, the key code from the blocked index is still readable.

For each PUF index: When 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.

41.2.7 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 41.2.6](#).

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, which is indicated by busy asserted. Before the system starts issuing commands, it waits for initialization to be finished. Use the function `wait_for_init` for the waiting period.

The code includes a ZEROIZE function. It can be used if software detects a reason to delete sensitive data, for example, when the PUF controller functions return an ERROR status.

[Table 210](#) defines the parameters of the functions. [Table 211](#) defines the data access functions. The system must supply them.

Key formats are defined in section [Section 41.2.6.6](#). It is assumed that the data and key are stored in memory in this format.

Table 210. 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 must be stored in some kind of NVM. When ACdata is requested, it must 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 must be stored in some kind of NVM. When KCdata is requested, it must 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 211. 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.

41.2.7.1 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  
}
```

41.2.7.2 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  
}
```

41.2.7.3 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, write AC to PUF
```

```
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
}
```

41.2.7.4 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
}
```

41.2.7.5 Pseudocode set key function

```
status set_user_key(KCdata, KeyIndex, KeyData) {
    // 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(KeyData)
```

```

// 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 to PUF and read KC from PUF
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
}

```

41.2.7.6 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 to PUF and read key from PUF
    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  
}
```

41.2.7.7 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  
}
```

41.3 AES engine functional details

The AES engine supports 128-bit, 192-bit, or 256-bit keys for encryption and decryption operations.

41.3.1 Features

- Encryption and decryption of data.
- Secure storage of AES key that cannot be read.
- AES engine peak performance of 0.5 bytes/clock cycle.
- AES engine supports 128-bit, 192-bit, or 256-bit key-in:
 - Electronic code book (ECB) mode
 - Cipher block chaining (CBC) mode
 - Counter (CTR) mode
- The AES engine is compliant with the Federal Information Processing Standard (FIPS), 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 the AHB master for data movement. Use the AHB master only 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.

41.3.2 Basic configuration

- AES functionality is combined with the SHA block, referred to as SHA-AES. For clock and reset connection programmability information, see the SHA basic configuration (see [Section 41.4.2](#)).
- For AES registers, refer to SHA-AES register description (see the section "Hash-AES peripheral (HASHAES)" in the Register description ([Ref. 5](#))).
- AES block shares same base address as SHA block.

41.3.3 General description

AES is a block cipher, and so encrypts and decrypts 128-bit blocks of data provided by the user. The AES block is bidirectional, that is, it can encrypt plaintext into ciphertext, or it can decrypt ciphertext from ciphertext.

In electronic code book (ECB) mode, the application provides a 128-bit input block, and the AES engine transforms that into a 128-bit output block. Subsequent blocks of 128 bits are treated independently from one another.

The AES engine supports two other cipher modes (CBC and CTR). There, the application first provides an IV of 128 bits, and then provides 128-bit data blocks.

For CBC encryption, the application will XOR the current plaintext block with the previous ciphertext block (or IV for the first block), and then run the result through to AES core to get the current ciphertext block. For CBC decryption, it's happening the other way around: the application runs the current ciphertext block through the AES core, and then XOR that with the previous ciphertext block (or IV for the first block) to get the current plaintext block. The chaining is obvious in this mode of operation.

For CTR encryption and decryption, the random nonce gets monotonically increased for each block. That updated nonce runs through the AES core. The result is XORed with the plaintext to get the ciphertext or is XORed with the ciphertext to get the plaintext.

The AES engine has an output/digest buffer of 256 bits. It also has two 512-bit buffers. It uses the first buffer to hold the AES key, the IV of CBC or {nonce + counter} of CTR, and the message block itself. The second buffer may be the output and a cache of the initial message for CBC decryption. CBC decryption must XOR the output of the AES core with the previous ciphertext block to get the plaintext for the current ciphertext block, so the current ciphertext block is temporarily stored here to allow for the next ciphertext block to be copied in.

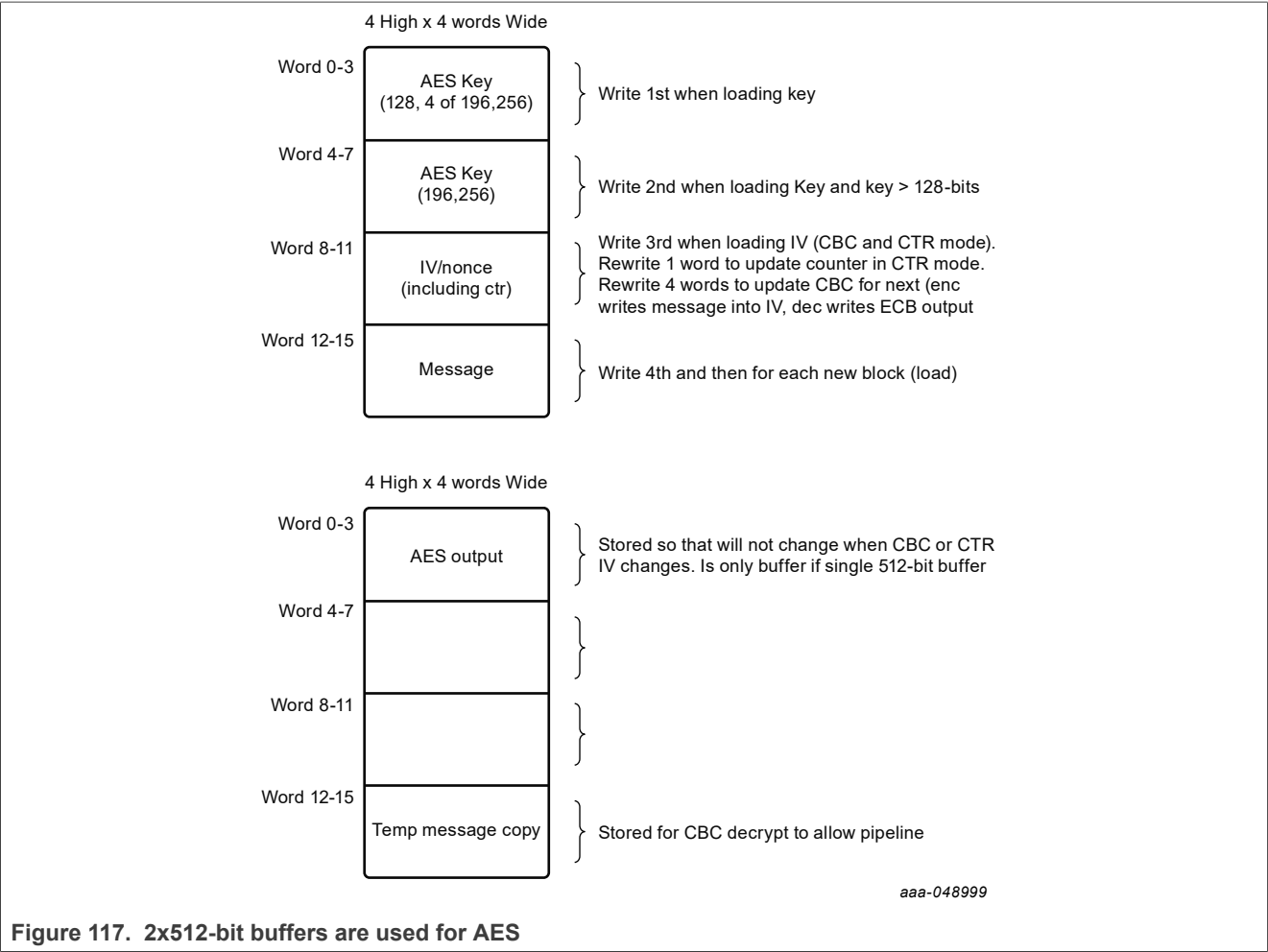


Figure 117. 2x512-bit buffers are used for AES

41.3.4 Using the AES engine

1. When starting a new operation, write the CTRL register NEW bit to initialize the engine. Choose whether the AES engine is the AHB master by setting the MASTER bit in the MEMCTRL register.
2. The AES engine uses a 128-bit, a 192-bit, or a 256-bit key, depending on the AESKEYSZ field in the CRYPTCFG register. Hardware (PUF) or software can supply the key.
 - If PUF supplies the key, see [Section 41.1.1](#), which describes PUF key loading for AES.
 - If software is selected to supply the key, write the 4, 6, or 8 keyword values into INDATA. These word values are placed in the correct place in buffer1. The STAT register NEEDKEY is 1 until it is completed.
3. The software-defined keys are not retained during power-down or deep-power down. They must be reloaded on reset. PUF keys are retained during power-down but not during deep-power down.
4. If a cipher mode is selected (rather than just ECB), write the IV/nonce using four words 128 bits. These words are placed in the correct place in buffer1. The STAT register NEEDIV is 1 until it is completed. It can be the AHB master, if it is enabled.
5. Read in the next 128-bit block of plain text or ciphertext.
 - If the AES engine is the AHB master, it actively reads in the four words.
 - If the AES engine is programmed to use DMA or the processor, the corresponding one is notified to provide the four words.
6. When the four message words are written, the encryption or decryption starts.
 - If a cipher mode is used, the block that is processed corresponds to the rules of that mode.
7. On completion, the data is ready in the OUTDATA0 until OUTDATA3 words. The following steps allow for load next and the read out of data:
 - If the AES engine is the AHB master, and the count is not 0, it loads in the next four words first. It allows starting to process the next block before the previous digest has been read out, to save time. The processor or DMA may also do it.
 - If the AES engine is steered using DMA, it triggers the reading of the four words. Else, the processor reads out the processed block via an interrupt.

41.3.5 AES performance

When using 128-bit keys, the AES block takes $33 + 2$ cycles for each block to encrypt. Using a 192-bit key adds six cycles. Using a 256-bit key adds twelve more cycles.

To decrypt when using 128-bit keys, the AES block takes $43 + 2$ cycles for each block. Using a 192-bit key adds six cycles. Using a 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 the first four word message data and the final output copying the final four words. All other input data and output results are pipelined. So, they do not add to the cost unless the application or DMA is slow.

41.4 HASH functional details

41.4.1 Features

- Performs SHA-1-based and SHA-2(256)-based hashing.
- Used with HMAC to support a challenge/response or to validate a message.

41.4.2 Basic configuration

- The priority for the SHA engine can be set in PRI_SHA bit in AHBMATPRIO register (see the section "AHB matrix priority control register (AHBMATPRIO)" in the Register description ([Ref. 5](#)).

- The SHA engine can be reset using the HASH_AES_RST bit in PRESETCTRL2 register (see the section "Peripheral reset control 2 (PRESETCTRL2)" in the Register description ([Ref. 5](#)).
- The clock to the SHA engine can be controlled using the HASH_AES bit in AHBCLKCTRL2 register (see the section "AHB clock control 2 (AHBCLKCTRL2)" in the Register description ([Ref. 5](#)).

41.4.3 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 necessary can be processed. The last block must be formatted per the SHA model:

1. The last data block must contain 447 bits or less. If more, 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 is a 1 followed by 447 zeroes. The remaining 64 bits contain the length of the whole message including the last block.

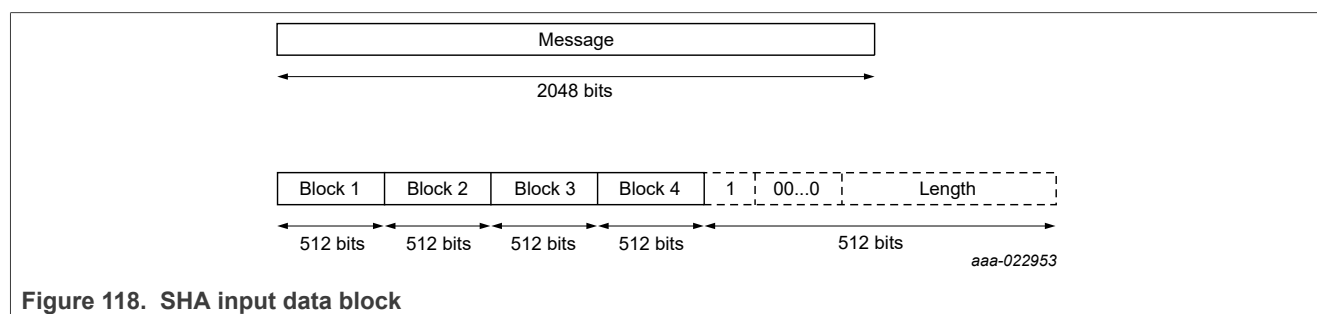


Figure 118. SHA input data block

The Arm processor uses little-endian. So, 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 the processor (or DMA) reads it as a word, is reversed into "dcba". When the input data is provided in little-endian format, the hash block swaps them to process correctly.

41.4.4 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. However, the LOCK register and the ID at offset 0xFFC can be read at any time.

41.5 True random number generator (TRNG) functional details

The true random number generator (TRNG) is a hardware accelerator module that generates a 512-bit entropy as needed by an entropy-consuming module or by other post-processing functions. A typical entropy consumer is a pseudo random number generator (PRNG) that can be implemented to achieve both true randomness and cryptographic-strength random numbers using the TRNG output as its entropy seed. The PRNG is not part of this module. The entropy generated by a TRNG is intended for direct use by functions that generate secret keys, per-message secrets, random challenges, and other similar quantities used in cryptographic algorithms. In each of these cases, it is important that a random number is difficult to guess or predict. It is important that a random number is at least as difficult to predict as it is difficult to break the cryptographic algorithm with which it is being used. This stringent requirement is difficult to fulfill if the entropy source from a TRNG contains bias and/or correlation. To increase the trustworthiness/quality of the generated random data, PRNGs are often used to post-process the output of a TRNG. This document describes only the TRNG design functionality and usage.

Before entropy can be obtained from the TRNG, it must be initialized and instantiated in a particular mode by setting the appropriate TRNG registers.

41.5.1 Entropy accumulation

Before entropy can be obtained from the TRNG, it must be initialized by setting the appropriate TRNG registers.

Upon initialization, the generation of the first block of the entropy starts. It is done once.

Read the entropy from the TRNG and start the generation of the next block of the entropy. This action is taken many times. It is the normal flow of operation.

41.5.2 Initialization

At power-on reset (POR), the TRNG resets to programming mode. It does not generate the entropy until it is out of programming mode (in run mode) and access to entropy registers has been enabled.

To write to most TRNG registers, the MCTL register must be initialized in programming mode as described in the "TRNG Register Descriptions" section.

41.5.3 Normal usage

Here is an example program flow of using the TRNG.

1. After POR, the TRNG is reset to programming mode with the OK-to-stop bit set ($MCTL[TSTOP\_OK] = 1$). The TRNG must be put in run mode for the entropy generation to start ($MCTL[PRGM] = 0$). Additionally, to have access to the entropy registers and other critical TRNG registers, the TRNG access bit must be set ($MCTL[TRNG\_ACC] = 1$). Using the default self-test limits that exist after bootup, the entropy valid bit can be polled until asserted ($MCTL[ENT\_VAL] = 1$). Alternatively, if using the interrupt, the interrupts are enabled via the INT_MASK register, and the ipi_rng_int_b is asserted when $MCTL[ENT\_VAL] = 1$.
2. After the polling completes, the 512-bit entropy that the TRNG generates can be read. The values can be read in any order from entropy register 0 to register 15 (ENT0 to ENT15). After reading ENT15, the old entropy value is reset and a new entropy value is generated.
Note: Reading ENT15 always resets the entropy. So, it must always be read last.
3. To handle reading the entropy values when the entropy valid interrupt is triggered, you can poll again for the new entropy value or you can use the interrupt status register.
4. The interrupt can be masked or cleared as necessary (see the interrupt status register description).
5. To change the self-test limits, the seed counters, how fast the entropy is generated, and how the entropy is sampled, see the register description section. In particular, see the TRNG frequency count minimum limit register (FRQMIN), the seed control register (SCML), the 6 statistical run length registers (RSCR1L - RSCR6L), and the poker range register (PKRRNG).
6. When in run mode, the entropy is regenerated automatically after ENT15 is read. To stop the TRNG or access TRNG registers at any point while in running mode, you can always set $MCTL[TRNG\_ACC] = 0$. Setting the TRNG back to programming mode ($MCTL[PRGM] = 1$) also achieves the purpose of stopping entropy generation.

41.6 PRINCE real-time encryption or decryption details

41.6.1 Functional details

The MCX W23 supports four regions for encryption and decryption, referred to as crypto regions (see [Figure 119](#)). Each crypto region resides at a 256 kB address boundary within the flash. For the 1 MB flash size in MCX W23, the four crypto regions are each 256 kB in size. Each crypto region is divided into 8 kB subregions, which can be individually enabled.

Each crypto region has a dedicated 128-bit key. 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 41.2.5](#) for more details.

In [Figure 119](#), each of the four crypto regions of flash is shown, including its base address and size. Also shown are the 32 subregions for each region, represented as a 4x8 grids of blocks. The highlighted subregions are marked with "c" are "crypto" enabled. It means that they are enabled for encryption and decryption. This diagram gives an example of how various subregions can be configured. Enabled subregions are not required to be contiguous. SKey# shown are the keys that PRINCE uses when encrypting or decrypting the data in the subregions of the crypto regions.

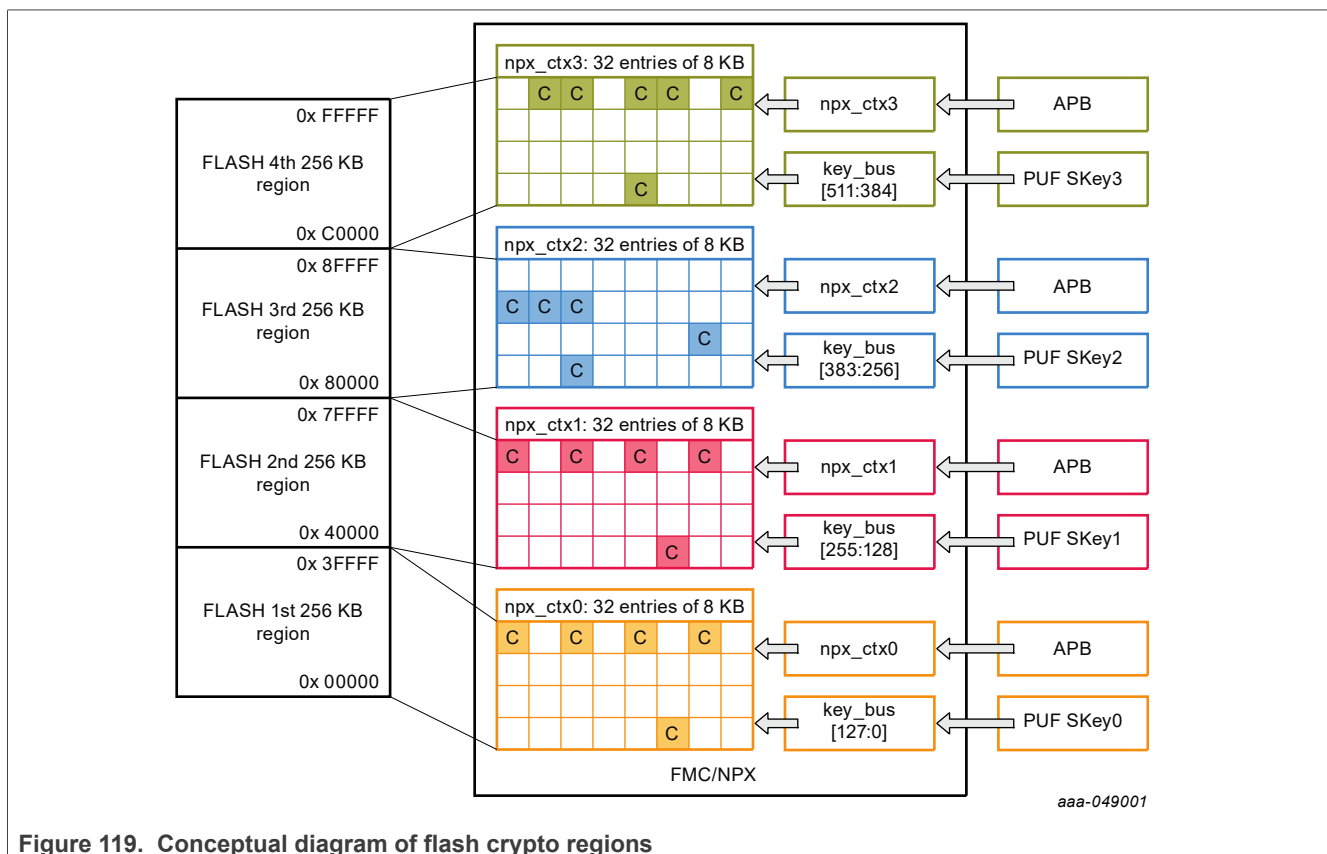


Figure 119. Conceptual diagram of flash crypto regions

PRINCE registers are retained during deep-sleep and power-down with CPU retention. However, without CPU retention and deep-power down, they are not retained during power-down.

41.6.2 Usage notes

The On-the-Fly (OTF) PRINCE XEX engine operates with the internal flash memory controller. The PRINCE XEX engine is directly connected to the flash command, address, read data, and write data interfaces. A private key bus (PKB), which shares the peripheral bus interface of the FMC, provides the PRINCE XEX logic with the necessary keys. The keys are transported and stored in an obfuscated form. The PRINCE block cipher design can do a complete 64-bit block size encrypt or decrypt in one cycle.

42 Cryptographic accelerator (CASPER)

The CASPER peripheral is available on all MCX W23 devices. The cryptographic accelerator (CASPER) engine provides acceleration of asymmetric cryptographic algorithms.

When the CASPER is used with hardware blocks for hashing and symmetric cryptography, a potential performance boost can be achieved.

Supported crypto functions are implemented in the software development kit (SDK) and the MBED TLS examples use the CASPER peripheral for computations. The CASPER interface uses SRAMX_0 (0x1400 0000 to 0x1400 0FFF) and SRAMX_1 (0x1400 1000 to 0x1400 1FFF).

42.1 CASPER features

When relying on just the MCU without CASPER assistance, the Arm CM33 must perform all the computational processing, which can slow down some applications with intensive processing requirements. Under these conditions, CASPER improves performance and frees up the Arm CM33 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 and 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 2 times faster than the Arm CM33. For a performance comparison for the various functions, see [Table 212](#):

Table 212. Performance comparison for elliptic curve calculations

Operation	Algorithm	Software only version Cortex-M33 at 32 MHz	With CASPER acceleration Cortex-M33 at 32 MHz	Performance improvement
Signing	ECDSA-secp256r1	1.67 sign/s	3.00 sign/s	1.79 times
Verification	ECDSA-secp256r1	1.00 verify/s	2.67 verify/s	2.67 times
Key exchange	ECDHE-secp256r1	1.00 handshake/s	2.00 handshake/s	2.00 times
Key exchange	ECDH-secp256r1	1.67 handshake/s	3.33 handshake/s	1.99 times

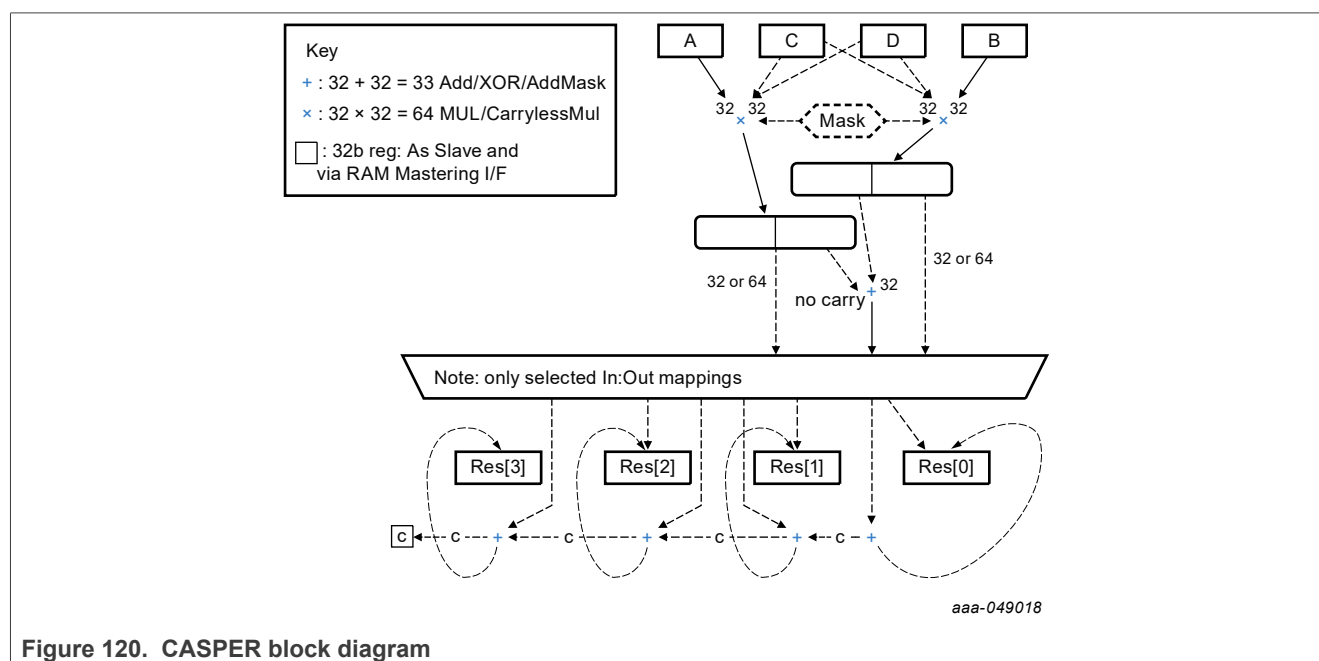
42.2 CASPER operation

The CASPER module incorporates:

- 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 64 x 64 multiplication. It is much faster than a separate adder.
- Four (RES[0-3]) result registers which can be used with four adders, to do add mask and XOR operations.
- Special access to two RAMs (up to 4 kB each) in parallel.
 - The block can access these two RAM banks simultaneously, allowing for two 64-bit operations to happen in parallel.

- Interleaving RAMs (that is, one for the even words and one for the odd words) allows 64-bit word pairs to be accessed simultaneously. SYSCON.CASPER_CTRL is used to enable the interleaved addressing on RAMX0 and RAMX1. However, when using these bits, it must be kept in mind that the RAMX interface for the CPU is also affected.
- A clock enable model is supported. So, when the speed of the clock is faster than the pipe, the clock enables support for an MCP (multicyclic path) approach to completion of the operations.

Figure 120 shows a conceptual representation of how CASPER operations are performed through the data and result registers and how the mask is applied.



CASPER uses SRAM bank X (8 kB 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 [Figure 121](#).

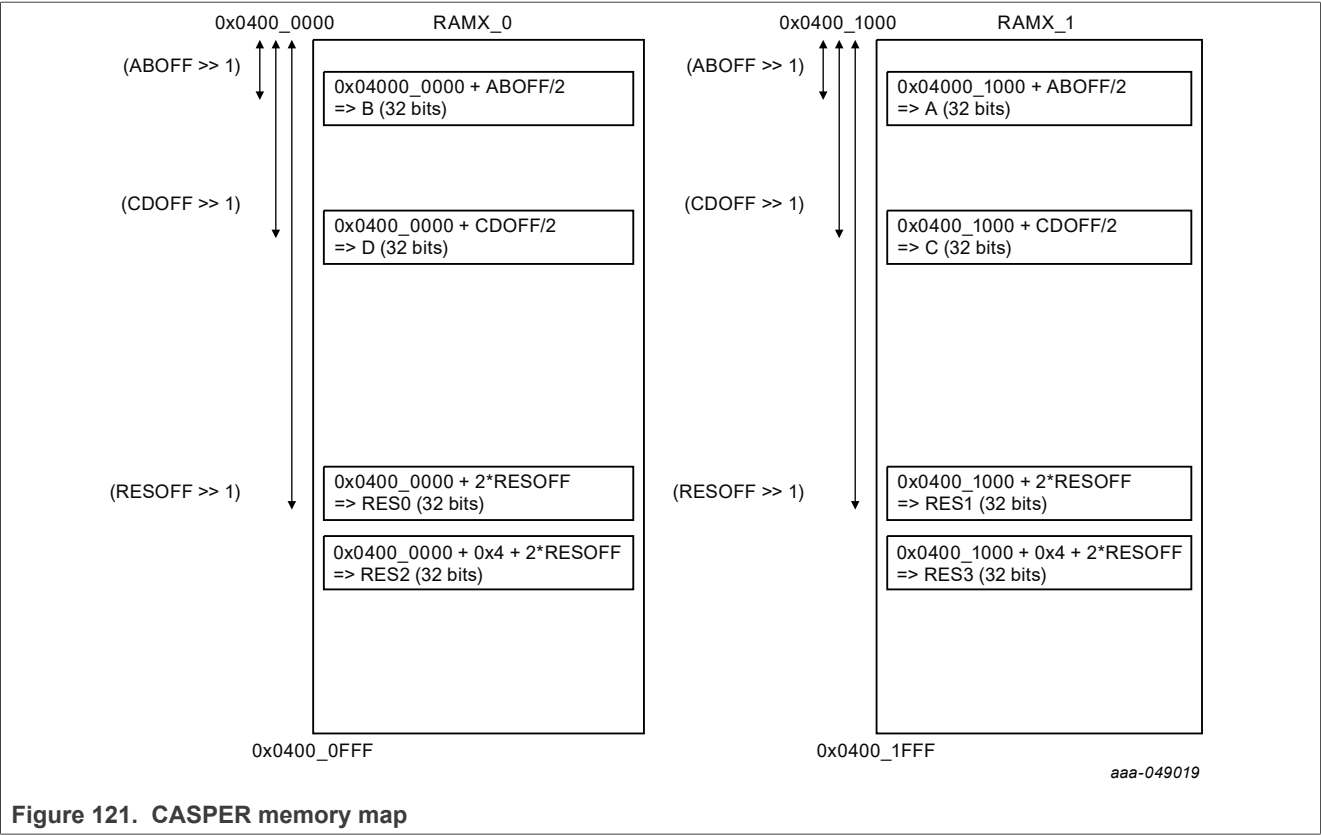


Figure 121. CASPER memory map

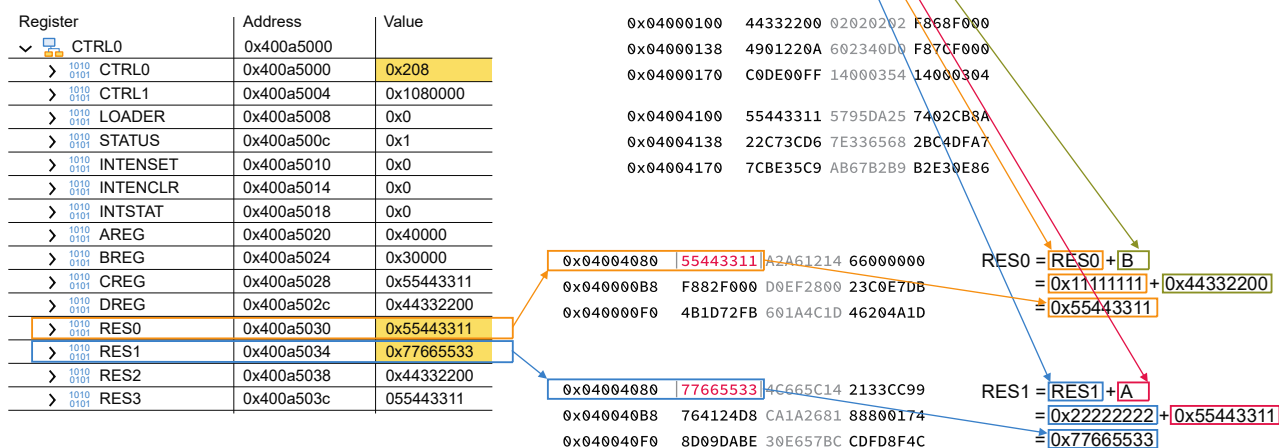
[Figure 122](#) shows how CASPER registers are stored and used in memory:

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();
```



aaa-049020

Figure 122. CASPER example

Based on this example:

- A and B are stored at (ABOFF >> 1):
 - B is stored at 0x0400_0000+(ABOFF >> 1)
 - A is stored at 0x0400_1000+(ABOFF >> 1)
- C and 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):
 - RES0 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

42.2.1 CASPER coprocessor operation

The ARMv8-M architecture (Arm CM33) provides a coprocessor interface that allows access to CASPER via move from coprocessor to register (MCR) and move from register to coprocessor (MRC) opcodes. Through this interface, up to two registers can be transferred between the Arm CM33 core and CASPER.

Upon submitting the data and/or opcodes to a coprocessor, the Arm CM33 can continue executing other tasks while the coprocessor performs its 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, 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 $CRm = 8$, which can be computed as $0x60 \gg 2 = 0x18$, where the lower nibble goes into CRm and the upper nibble into CRn .

The MCRR and MRRC instructions allow access to pairs, including CTRL0/CTRL1, A/B, C/D, RES0/RES1, RES2/RES3. The lower index of the pair is used in CRm , with MCR/MRC.

42.2.2 CASPER AHB operation

For applications to be able to access the control and data/results registers as necessary, a bus slave (AHB) and Arm M33 CP interface is provided. It can set up the operation, the iteration count, and the offset registers (offsets into the RAMs). It 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.

If necessary, it may access the data and result registers.

42.2.3 CASPER modes

Table 213. Casper AHB operations

Mode	Name	Description	Comments
0x01	MUL6464_NOSUM	Walking 1 or more of J loop, doing $w[j] = ab * cd[j]$, based on $64 \times 64 = 128$.	Writes out results, but does not read in to add.
0x02	MUL6464_SUM	Walking 1 or more of J loop, doing $c, w[j] = w[j] + ab * cd[j]$, based on $64 \times 64 = 128$.	Sums by reading result word ($w[j]$) and adding before writing back. 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 $c, w[j] = w[j] + ab * cd[j]$, based on $64 \times 64 = 128$, sum all of w.	Reads all of w, including the MSWs ^[1] , can have a carry in the carry bit Includes 1st loop half of CIOS ^[2] multiply use (before reduce).
0x04	MUL6464_REDUCE	Walking 1 or more of J loop, doing $c, w[j - 1] = w[j] + m * cd[j]$, based on $64 \times 64 = 128$, but skip 1st write. Note: <i>m is in ab and is $w[0] * Np$.</i>	- Does not compute m into ab, you need to this first. - The reduction pass of a CIOS^[2] double J loop. So, it writes to the preceding result doubleword, 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 $N \setminus Q$ as a 64-bit value). - Similar to the MUL6464_FULLSUM operation, except it skips the first write so it can write to the previous.
0x08	ADD64	ADD with ABOFF, and in/out RESOFF base on $c, r = r + a + c$.	Uses 64 bits at a time, producing 65-bit output. Final carry in the carry bit.
0x09	SUB64	SUBTRACT with ABOFF, and in/out RESOFF base on $r = r - a$.	Uses 64 bits at a time, and if a is larger, final borrow is implicit, but carry bit set if borrowed. Solved using $r = r + (\sim a + 1)$.
0x0C	RSUB64	SUBTRACT with ABOFF, and in/out RESOFF base on $r = r - a$.	Uses 64 bits at a time; if a is larger, final borrow is implicit, but carry bit set if borrowed. Solved using $r = r + (\sim a + 1)$.

Table 213. Casper AHB operations...continued

Mode	Name	Description	Comments
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 0 s, 64 bits at a time.	

[1] MSW = most significant word; LSW = least significant word

[2] CIOS = coarsely integrated operand scanning

43 Debug subsystem

Debugging is supported through the serial wire debug (SWD) interface of Arm. It enables 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 that the reader has prior knowledge of the SWD interface of Arm and coresight debug and trace technology.

Note: JTAG is only used on this device for boundary scan and production test. It cannot be used for debugging purposes, so it is not described in this chapter.

43.1 Features

- Supports Arm serial wire debug mode for CPU0.
- Trace port provides Cortex-M33 CPU instruction trace capability on CPU0. Output via a serial wire viewer.
- Direct debug access to all memories, registers, and peripherals.
- No target resources are required for the debugging session.
- Breakpoints: CPU0 includes eight instruction breakpoints.
- Watchpoints: CPU0 includes four data watchpoints that can also be used as triggers.
- Instrumentation trace macrocell (ITM) allows additional software-controlled trace for CPU0.

43.2 Basic configuration

This device supports the serial wire debug (SWD) interface of Arm. SWD is the default function for pins PIO0_11 (SWCLK) and PIO0_12 (SWDIO) after a reset. To use SWO, it must be enabled in the application code by selecting the SWO function on either PIO0_15 or PIO0_17.

The ROM controls debug by a remote host. It is only enabled when the device configuration permits it and when, to initiate a debug session, the correct protocol is followed. If the device has been configured for debug authentication, a debug session must be initiated following the correct authentication sequence. For details about the mailbox protocol, see [Section 43.5](#). For details about debug authentication, see [Section 43.6](#).

43.3 Pin description

[Section 43.3](#) 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 214. 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_15 or PIO0_17	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.

To enable SWO output on GPIO PIO0_15 (FUNC1) or PIO0_17 (FUNC1), the following setups are required:

1. To enable the trace clock divider, Write 0x0 to TRACECLKDIV (see the section "TRACE clock divider (TRACECLKDIV)" in the Register description ([Ref. 5](#)).
2. If the clock to the IOCON block is not already enabled, write to AHBCLKCTRLSET0 (see sections "AHB clock control set register" (AHBCLKCTRLSET0 to AHBCLKCTRLSET2) in the Register description ([Ref. 5](#)). To access any IOCON registers, the clock must be enabled.
3. Set the FUNC1 value in the IOCON register corresponding to PIO0_15 or set the FUNC1 value in the IOCON register corresponding to PIO0_17.

43.4 Debug subsystem functional description

This section describes the hardware elements of the debug subsystem and defines the command protocols they use. For descriptions of how the host debug system interacts with the debug subsystem during debug sessions with and without debug authentication, see [Section 43.6](#) and [Section 43.5](#).

43.4.1 Debug subsystem

[Figure 123](#) shows the top-level debug ports and connections in the MCX W23. The designated AP is used to specify access port. Blocks SWJ-DP and DM-AP are always enabled and are accessible through the SWD interface. Remaining blocks are enabled/disabled under hardware state machine and software control.

- DAP
Debug access port with a serial wire port (SWJ-DP), which interprets the data coming in and routes it to the appropriate access port (AP).
- CPU AP
Debug access port for Cortex-M33 core instantiated as CPU.
- 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.

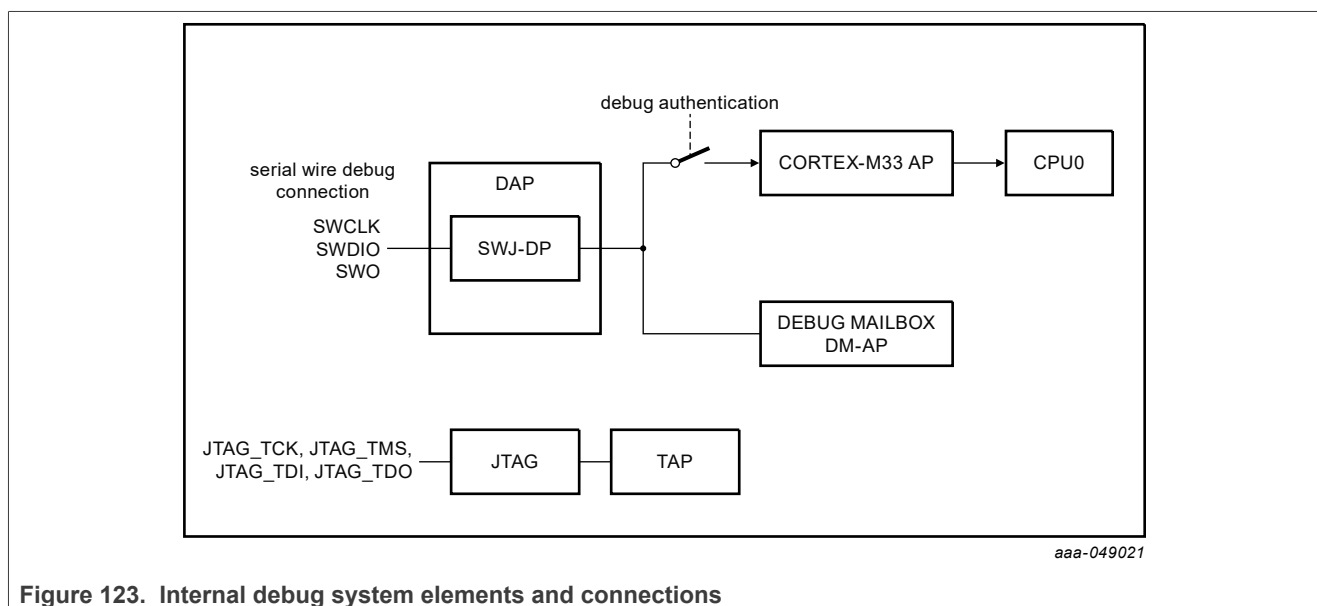


Figure 123. Internal debug system elements and connections

43.4.2 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. [Table 214](#) describes external I/O pins that interface with the DAP. The DAP block is always enabled. However, the I/O pins that provide access to the SWD signals may be used for other functions under control of software (see [Section 15](#)).

43.4.3 CPU0 AP

The DAP for CPU0 is disabled during a power-on reset or during assertion of the reset pin. If the correct debug initiation procedures are followed, the ROM enables it. If the DAP is used, the debug enablement protocol can be used to initiate a debug session. If debug authentication is required, see [Section 43.6](#). The debug authentication process allows control of the DBGEN, NIDEN, SPIDEN, and SPNIDEN signals that the Cortex-M33 generates. It is described below.

DBGEN

Invasive debugging of the non-secure domain, as TrustZone for Arm8-M defines it.

- Breakpoints and watchpoints to halt the processor on specific activity.
- A debug connection to examine and modify registers and memory and to provide single-step execution.

NIDEN

Non-invasive debugging of the non-secure domain, as TrustZone for Arm8-M defines it.

- A collection of information on instruction execution and data transfers.
- Deliver trace in real time to off-chip tools to merge data with source code on a development workstation for future analysis.

SPIDEN

Invasive debugging of the secure domain, as TrustZone for Arm8-M defines it.

SPNIDEN

Non-invasive debugging of the secure domain, as TrustZone for Arm8-M defines it.

43.4.4 Debugger mailbox AP

The debugger mailbox (DM) AP is a register-based mailbox that the CPU0 and the device debug port DP of the MCU can access. 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 the ROM on CPU0. This port is used to implement the NXP debug authentication protocol. For a description of the protocol that is used to initiate a debug session from a host debug system, see [Section 43.6](#).

43.4.5 Reset handling

The debug domain (DP, CM33-AP, DM-AP) is reset upon POR (Power-On Reset) or pin reset (assertion of RESET_N). On other resets, the debug domain retains its state and the defined breakpoints, watch points, and so on, survive even when the debugging tool issues a reset to the device. The matrix in [Table 215](#) describes the various reset types and shows which associated domains/components get reset by each type. "Rst" indicates that the the associated reset source resets the subdomain, and "Act" indicates that it remains active and does not change state in response to that reset source.

The DAP for CPU0 is disabled during a power-on reset or assertion of the reset pin, and enabled by the ROM if/when the correct debug initiation procedures are followed. If the DAP is not being used, the debug enablement protocol can be used to initiate a debug session. If Debug Authentication is required, see [Section 43.6](#).

Table 215. Resets

Chip reset/wakeup for the following domains and components:		POR (Power-on Reset)	RESET_N	BOD RESET	SYSTEMRESET	Debug mailbox	WDT RESET	SWR RESET	DPDRESET_WAKEUPIO	DPDRESET_RTC	DPDRESET_OSTIMER	Wake-up from Power Down	Wake-up from Deep Sleep	Wake-up from Sleep
always-on domain	PMC	rst	rst	rst	act	act	act	act	act	act	act	act	act	act
	OSTIMER	rst	rst	rst	act	act	act	act	act	act	act	act	act	Act
	RTC	rst	act	rst	act	act	act	act	act	act	act	act	act	Act
system domain	IOCON	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act	act
	SYSCON	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act	act
	SYSCON (debugger mailbox enable)	rst	rst	rst	act	act	act	act	rst	rst	rst	act	act	act
	PUF key management	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act	act
	GINT0	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act	act
	Flexcomm_0	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act	act
core domain	SRAM retention memory	rst	rst	rst	act	act	act	rst	act	act	act	act	act	act
	SRAM memory	rst	rst	rst	act	act	act	rst	rst	rst	rst	rst	act	act
	Flash memory	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	Flash controller	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	PRINCE	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act	act
	WWDT	rst	rst	rst	rst	rst	act	rst	rst	rst	rst	rst	act	act
	UTICK	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	MRT	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	SCT	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	debug mailbox	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	Arm CM33	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act	act
	CASPER	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	PUF	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	TRNG	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	CRC	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	HASH	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	secure AHB ctrl	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act	act

Table 215. Resets...continued

Chip reset/wakeup for the following domains and components:		POR (Power-on Reset)	RESET_N	BOD RESET	SYSTEMRESET	Debug mailbox	WDT RESET	SWR RESET	DPDRESET_WAKEUPIO	DPDRESET_RTC	DPDRESET_OSTIMER	Wake-up from Power Down	Wake-up from Deep Sleep	Wake-up from Sleep
core domain (continued)	DMA	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	ANACTRL	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	CTIMER	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	Flexcomm (all except 0)	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	INPUTMUX	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	GPIO	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	PINT	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	SECURE GPIO	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	SECURE PINT	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	PLU	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
analog components	DC-DC	rst	rst	rst	act	act	act	rst	rst	rst	rst	rst	act	act
	BIAS	rst	rst	rst	act	act	act	rst	rst	rst	rst	rst	act	act
	BOD1	rst	rst	rst	act	act	act	rst	rst	rst	rst	rst	act	act
	BOD2	rst	rst	rst	act	act	act	rst	rst	rst	rst	rst	act	act
	LDO_AO	rst	act	act	act	act	act	act	act	act	act	act	act	act
	LDO_MEM / LDO_DIG	rst	rst	rst	act	act	act	rst	rst	rst	rst	rst	act	act
	32.768 kHz Xtal	rst	rst	rst	act	act	act	act	act	act	act	act	act	act
	32 MHz Xtal	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	32.768 kHz FRO	rst	act	act	act	act	act	act	act	act	act	act	act	act
	192 MHz FRO	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	rst	act	act
	1 MHz FRO	rst	rst	rst	act	act	act	rst	rst	rst	rst	rst	act	act

43.4.6 Mailbox commands

This section describes the response and request message formats and available mailbox commands, and their associated parameters.

43.4.6.1 Request

The first word transmitted in a request is a header word containing the command ID and number of following data words. The command packet is sent to the device by writing 32 bits at a time to the REQUEST register. When sending command packets greater than 32 bits, the debugger must read an ACK_TOKEN in the RETURN register before writing the next 32 bits.

Following the header is the number of 32-bit words specified in the header.

Table 216. 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[];
};
```

DM-AP commands

[Table 217](#) lists the commands for the DM-AP. These commands are written to the REQUEST register. Except for the "Enter ISP mode" and "Start Debug Session" command, issuing an "Exit DM-AP" command to resume normal device boot flow normally follows the issuing of a command (or sequence of commands).

Table 217. DM-AP commands

Name	Command code	Parameters	Response	Description
start DM-AP (legacy command)	0x01	none	32-bit status	Causes the device to enter the DM-AP command mode before sending bulk erase commands. This command is provided for backward compatibility and does not have to be used.
reserved	0x02	none	32-bit status	reserved, returns 3.
bulk erase (legacy command)	0x03	none	32-bit status	Erase the entire on-chip flash memory (excluding protected flash regions).
exit DM-AP (legacy command)	0x04	none	32-bit status	Causes the device to continue normal debug flow.
enter ISP mode	0x05	dataWordCount: 0x1 data[0]: ISP mode enum • 0xffffffff - Auto detection • 0x1 - UART • 0x2 - I2C • 0x4 - SPI • others - reserved	32-bit status	Enter specified ISP mode. By default, the state of the ISP boot selection pins at reset time determines the ISP mode entry. Usually, this functionality is disabled through CMPA configuration before 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_CC_SOCU(_NS) fields in the PFR, as described in Section 43.6 .
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, such that suspect parts can be sent to NXP for FA/RMA testing. Support of this command can be restricted through the DCFG_CC_SOCU(_NS) field in the PFR, as described in Section 43.6 .
start debug session	0x07	none	32-bit status	When program control is in ROM memory context (that is, instructions are fetched from a ROM memory address range) during boot, the debug

Table 217. DM-AP commands...continued

Name	Command code	Parameters	Response	Description
				access is disabled irrespective of the device life-cycle state or DCFG_CC_SOCU(_NS) settings. This command instructs the boot code in ROM to clean up memory and peripherals (SysTick, UART, SPI, I2C, SPIFI, MPU, and SAU) that the boot code initializes. It also instructs the boot code to enter safe processor execution context for external debuggers to attach. On systems where there is no valid image in boot media, the program control enters ISP command handler loop, which is still in ROM memory context, and so debug access is disabled. As a result, to indicate to the ROM its intention of connecting to the debugger, an external debug system must use this command. 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 authentication start	0x10	none	dataWordCount: 0x12C or 0x22C data[]: debug authentication challenge	start debug authentication protocol ROM responds to debugger with a debug authentication challenge message
debug authentication response	0x11	dataWordCount: 0x12C data[]: DAR	32-bit status	debug authentication response

43.4.6.2 Response

The first word transmitted in a response is a header word containing the command status and number of following data words. The command response can be read 32-bits at a time through the RETURN register. The initial 32 bits contain the response header as shown in [Table 218](#). When reading a response greater than 32 bits, the debugger must 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 commands and response values (supported on LPC microcontrollers), 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 218. 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[];

};
```

Response codes for DM-AP commands

Table 219 lists the response codes for DM-AP commands. These commands can be read from the RETURN register. The upper 15 bits (bits [30:16]) of the response code only contain relevant data (that is, the data word-count related to the command) when the command is successful.

Table 219. DM-AP response codes

Return code (bits 15:0)	Description
0x0000	Command succeeded.
0x0001	Debug mode not entered. Returned if other commands are sent before the "Enter DM-AP" command.
0x0002	Command not recognized. Another command than the ones defined above was received.
0x0003	Command failed.

43.4.6.3 ACK token

- When a command has parameters, the debugger waits for the ACK_TOKEN (sent through the RETURN register) before sending the next 32-bit value.
- When the device has a response packet to send back to the debugger, the ROM code waits for the debugger to send an ACK_TOKEN (sent through the REQUEST register) before sending the next 32-bit value.
- The receiving end sets the upper 16-bits of ACK_TOKEN with the number of remaining words that are expected.
- Lower 16-bits are always set to 0xA5A5.

Table 220. 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 */
};
```

43.4.6.4 Error handling

If an overrun occurs from either side of the communication, the appropriate error flag is set in the CSW register of the DM-AP. When an overrun has occurred in that direction, the state machine hardware prevents further communication in any direction. So, if such an error occurs, the debugger must start with a new synchronization request to clear the error flag.

43.5 Debug session protocol

To interact with host debug systems over the SWD interface, the MCX W23 boot ROM implements a debug mailbox protocol.

The protocol has the 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

To attach the debugger in a predictable manner, the ROM provides three debug methods/mechanisms:

- Start debug session method
- Debug access trigger method
- Debug authentication mechanism

Debug authentication is disabled by default. It is set up (if necessary) during development or during device provisioning at the production phase of a product using the device. For full details of debug authentication, see [Section 43.6](#).

On MCX W23 parts, when program control is in ROM memory context (that is, 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_CC_SOCU(_NS) settings. This mechanism is referred as 'boot-ROM protection' in this document. So, the method to initiate a debug session varies, 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 a 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 to 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 in flash 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").

43.5.1 Debug session with uninitialized/invalid flash image or ISP mode

When the device boots, no valid image may be available in the boot media. If so, the ROM-based program control enters the in-system programming (ISP) loop. Debug access is disabled for security reasons. Another scenario is where the device may be placed in 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 that the state machine, which controls the debug mailbox commands is in a known state, the debugger may have to reset this logic. Resetting this logic 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 resynchronization and resetting the device, the debugger reads the CSW register. The debugger must wait until the device has completed the resynchronization process, indicated by reading a value of 0 from the CSW register (checking only the lower 16 bits of this 32-bit value).

To start a debug session and control the exchange of debug information, use the DM-AP commands. Following a successful initial resynchronization, communication from 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 217](#)). The debugger can read results of communications via the RETURN register. To ensure that transactions have completed successfully, the debugger must poll the RETURN register in the same manner as it polled the CSW following a resynchronization request. [Table 219](#) shows the response codes. To handle situations where debug is disabled due to boot-ROM protection, the debug system must use a specific command (start debug session). It must also 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 that any unwanted peripheral interrupts are disabled and secrets managed before enabling debug access. After enabling debug access, the ROM enters a while (1) loop.

When the start debug session and device reset have been successfully executed, the APs for CPU0 are accessible and can be used to set breakpoints, and so on, 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 RESYNCH_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 "RESYNCH_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 flashloader is normally loaded into RAM and used to program the application to be debugged and the required breakpoints in the application code set. When it is done, a SYSRESET_REQ command must be issued to ensure that the ROM fully executes (as happens in a deployed end application) before reaching the downloaded application.

43.5.2 Debug session with valid application in flash

In this scenario description, it is assumed that an application already in flash has not disabled debug. So, the CPU0 AP is visible and breakpoints can be set without the necessity to resynchronize the mailbox hardware or issue a debug session request. To simplify debug support implementations, the same methods described in [Section 43.5.1](#) may be used.

43.5.3 Debug session attaching to a running target

In this scenario, the device has booted and runs an application that has not disabled debug and the host system attempts to connect to that device without resetting it and with no intention to update flash. In this case, the CPU0 AP must be visible and breakpoints can be set without the necessity to resynchronize the mailbox hardware, issue a debug session request, or issue a reset.

43.5.4 Halting execution immediately following ROM execution

Traditionally, debug systems may set a vector catch at the reset vector to break code execution at this point. However, the MCX W23 ROM prevents the setting of such a vector catch. To allow the debug system to halt execution immediately after the ROM has completed preparations after a debug session request, a watchpoint 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 watchpoint to halt the core on read of address location 0x50000040.
2. If all data watchpoint comparators are occupied, back up one of the watchpoint settings and replace it with the above watchpoint location.
3. Reset the core and peripherals by setting the SYSRESETREQ bit in the AIRCR of the SCB.
4. Wait for 100 ms to allow ROM to reenale debug access.
5. Check the SHCSR register of the SCB to see the core halted due to data watchpoint.
6. If reading SHCSR register of the SCB times out or returns error response, it indicates that the ROM entered an ISP command handling loop due to an invalid image in the boot media.
 - Execute start debug session sequence described in [Section 43.5.1](#).
 - Wait for 10 ms.
 - Now, the CM33_AP must be enabled.
 - Read SHCSR of the SCB and issue a HALT to the CM33_AP.
7. Clear the data watchpoint added in step 1. If a watchpoint was set as described in step 2, restore the watchpoint configuration.

43.6 Debug authentication

The MCX W23 offers a debug authentication protocol that is used to authenticate the debugger and grant access to the device. The debug authentication is a challenge-response scheme that ensures that the debugger has the appropriate debug credentials required to use the debug interface and access-restricted parts of

the device. Access rights can be preconfigured and are loaded into the register upon successful debug authentication. Until the debug authentication has successfully completed, the secure part of the device is not accessible to the debugger. See [Section 43.6.2](#) for a discussion of the various access levels. The following diagram is a generic example of a debug authentication flow.

1. The vendor generates RoT key pairs and programs the device with SHA256 hash of RoT public key hashes before shipping.
2. The field technician generates a key pair and provides the public key to the vendor for authorization.
3. The vendor attests the public key of the field technician public key by signing it with one of the RoT private keys. In the created debug credential certificate, the vendor assigns the debug access rights for that field technician.
4. The end customer who has issues with a locked product takes it to the field technician.
5. The field technician uses valid credentials to authenticate to the device and unlocks the product for debugging.

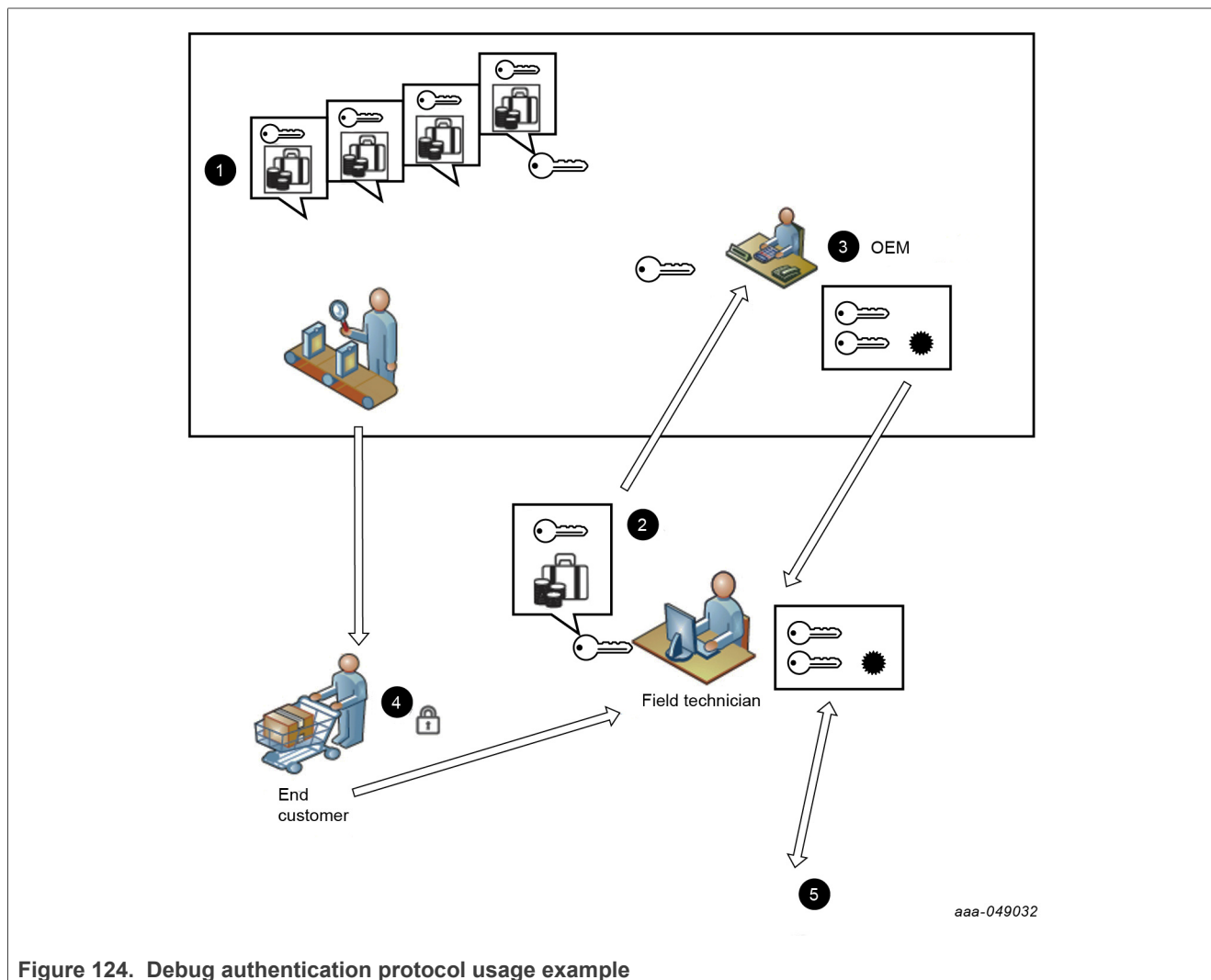


Figure 124. Debug authentication protocol usage example

The debug authentication protocol supports two versions of RSA key sizes. Version 1.0 supports RSA2048. Version 1.1 supports RSA4096. The vendor selects the version according to the security requirements. It means that, when vendors want to make debug authentication possible, they must choose the size of its RoT keys accordingly.

This protocol is divided into steps as shown in [Figure 125](#).

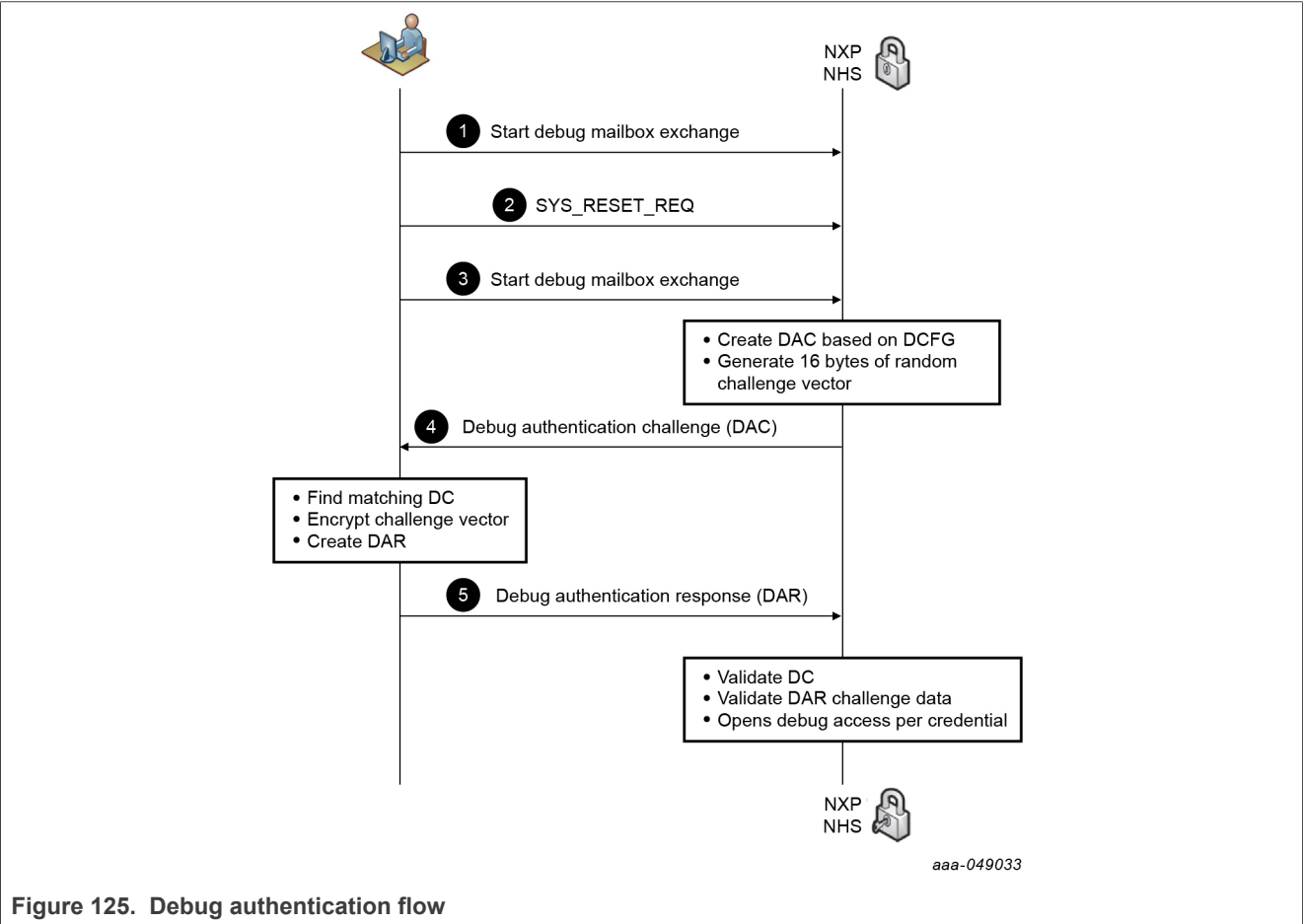


Figure 125. Debug authentication flow

43.6.1 Debug credential certificate (DC)

By prior construction, the debugger must already have a DCK (debug credential key). The public key part of this key pair is used to represent the identity of the debugger/field technician through the creation of a DC. It binds that public key to some usage attributes (which features the field technician must have access to). To comply with the trusted computing protocol, the root of trust key (RoT) of the vendor then authorizes/signs it. The total DC size is 940 bytes (v1.0) or 1708 bytes (v1.1).

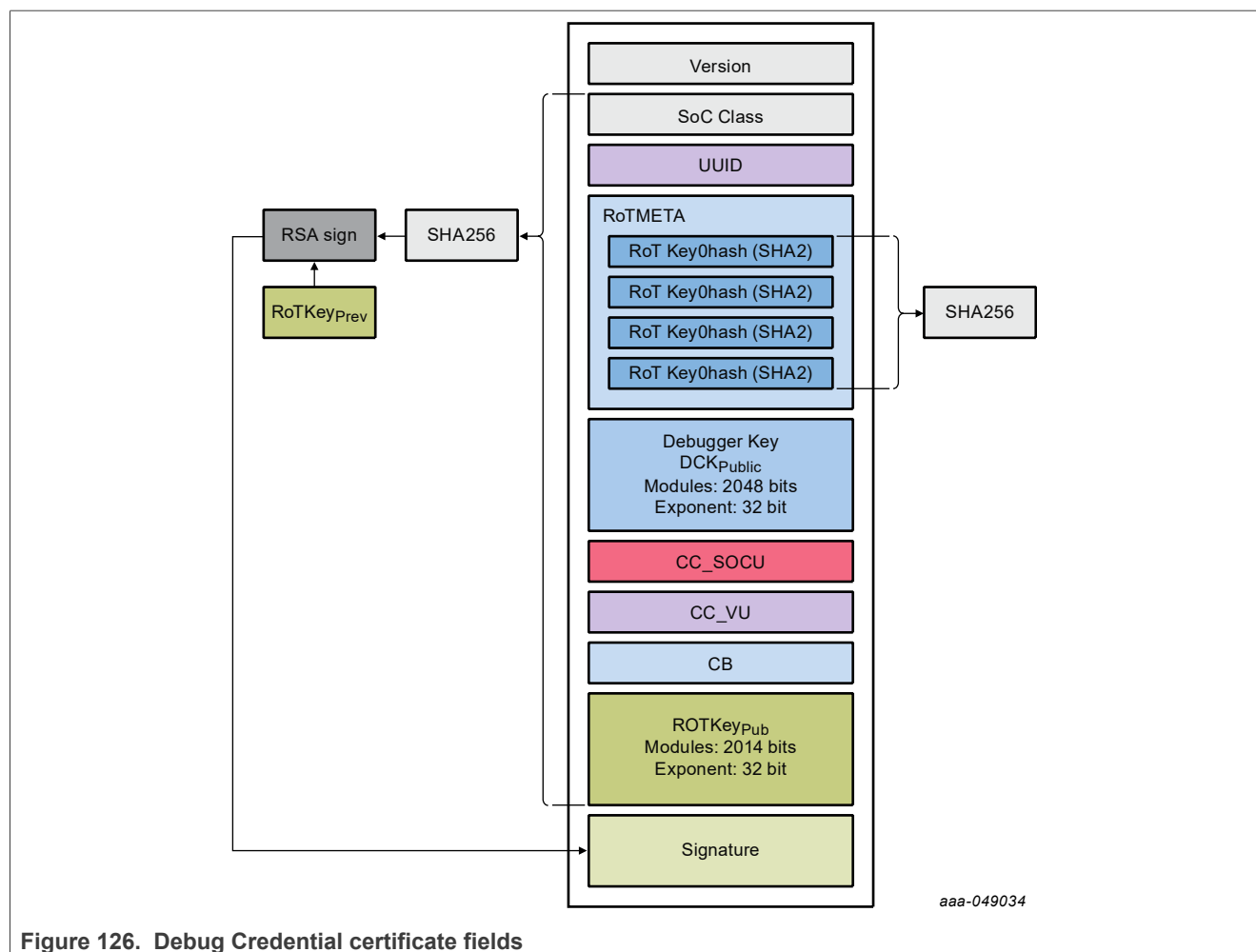


Figure 126. Debug Credential certificate fields

The data structure is represented as a packed binary concatenation of its component fields as shown in the list below:

- VER, The protocol version that credential applies to
4 bytes, value: 0x00010000 (v1.0) or 0x00010001 (v1.1).
- SoCC, the SoC class that this credential applies to
4 bytes, value for the MCX W23: 0x0001.
- UUID, the unique device identifier, checked only if this credential is used on a device configured for UUID matching.
If UUID matching is enabled in DCFG_CC_SOCU_(NS_)PIN on a particular device, it only accepts DCs that have the correct UUID.
The DC also works on any device in the whole SoC class where UUID matching is not enforced.
16 bytes.
- ROTMETA, the vendor RoT public keys that the device uses to verify the signature of this DC. It must map against the RKTH of the CMPA.
Four root public keys are supported to allow for key revocation.
On the MCX W23, SHA256 hashes of four RoT public keys must be set in this field.
128 bytes.
- DCK_{pub}, public key part from DCK.
260 bytes (v1.0): 2048-bit modulus and 32-bit exponent of public key.
516 bytes (v1.1): 4096-bit modulus and 32-bit exponent of public key.

- CC, credential constraints that the vendor has associated with this credential.
8 bytes; SoCU 4 bytes, VU 4 bytes.
SoCU, SoC specific credentials; On the MCX W23, only ten constraints are specified.
VU, vendor-specific constraints; must match the VENDOR_USAGE field in PFR, 2 bytes from the factory area (CMPA) and 2 bytes from the in-field area (CFPA). The latter can be used to revoke debug certificates.
- CB, credential beacon that the vendor has associated with this credential.
On the MCX W23, a nonzero value in this field delays opening the debug port. This feature can be useful if the user does not want the ROM to enable the debug ports immediately when the boot image starts executing. However, the application must do it at a certain point during its execution, where it can, for example, clean up sensitive information before opening the debug ports. It allows for always hiding the execution of certain parts of the application code. Also, the actual value of the beacons stored in the DEBUG_AUTH_SCRATCH register can be used for some user-defined purpose.
4 bytes.
- RoTKey_{pub}, root of trust vendor public key used for signing this certificate.
260 bytes (v1.0): 2048-bit modulus and 32-bit exponent of public key.
516 bytes (v1.1): 4096-bit modulus and 32-bit exponent of public key.
- SIG, A cryptographic signature by the RoT over the eight previous fields. Ensures that the vendor "blessed" the DC for the debugger to use.
256 bytes (v1.0) or 512 bytes (v1.1), as calculated below.

SIG(RoT_{priv}, HASH(DC :: 1 || DC :: 2 || ... || DC :: 8)).

- SHA256 is used for the hash function.
- RSASSA-PKCS1-v1_5 is used for SIG function.
 - The RSA signature scheme from PKCS1 specification v1.5 (RFC 2313).

From the debug authentication protocol perspective, what matters is that the debugger selects a debug credential that successfully authenticates, based on the constraints provided in the debug authentication challenge (DAC). It provides the required debug behavior postauthentication (for example, if to debug the secure world, with the desired credential beacon). If such a credential cannot be found, the debugger must report a corresponding error.

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.

43.6.2 Data stored in protected flash region (PFR)

The device side of the debug authentication process is already implemented in the ROM code. It does not require any further implementation. However, to define debug access rights, fields in the customer manufacturing programmable area (CMPA) at 0x1010_4000 - 0x1010_5FFF and the customer field programmable area (CFPA) at 0x1010_0000 - 0x1010_3FFF must be programmed. The ROM code use fields in these areas to enable the chosen access rights at the end of the debug authentication process.

DCFG_VER: Version of the debug authentication protocol that the device supports:

- Support is provided for debug authentication protocol version 1.0 and version 1.1.
- During the challenge stage, the packet is constructed as below:
 - Debug_auth_challenge.version.major = 0x1.
 - Report minor version based on CMPA.SECURE_BOOT_CFG.RSA4K value in PFR.
 - If RSA4K = b'0, debug_auth_challenge.version.minor = 0x0.
 - If RSA4K = b'1, debug_auth_challenge.version.minor = 0x1.

DCFG_ROTID: Root of Trust ID:

- Depending on the chip, the RoT public keys field for validating image authentication keys and debug credential keys.
- This field is 256-bit wide.
- The value of this field is stored in PFR CMPA.RKTH at 0x1010_4060 - 1010_407F.
- It is the SHA256 hash digest of hashes of four RSA public keys.
- $\text{CMPA.RKTH} = \text{SHA256}(\text{SHA256}(\text{RoTK0}_{\text{pub}}) \parallel \text{SHA256}(\text{RoTK1}_{\text{pub}}) \parallel \text{SHA256}(\text{RoTK2}_{\text{pub}}) \parallel \text{SHA256}(\text{RoTK3}_{\text{pub}}))$.
- $\text{RoTK0}_{\text{pub}} = 2048\text{-bit modulus} \parallel 32\text{-bit exponent}$.
- $\text{DCFG\_ROTID} = \text{CMPA.RKTH}$.

DCFG_CC_SOCU: Credential constraints:

DCFG_CC_SOCU is a bit mask that specifies debug access rights. It is derived from a combination of the PFR words CMPA.DCFG_CC_SOCU_DFLT, CMPA.DCFG_CC_SOCU_PIN, CFPA.DCFG_CC_SOCU_NS_DFLT, CFPA.DCFG_CC_SOCU_NS_PIN:

Lower half-words of these PFR words define the functionality.

Upper half-words of these PFR words are inverse of the lower halfword. Offers protection against glitch attacks.

The ROM code compares half-words before arriving at the final value. If there is a mismatch, the corresponding debug domain is disabled.

The DCFG_CC_SOCU_DFLT setting defines the default access rights for corresponding debug domains.

- If the bit is set, access is allowed. Otherwise, access is denied.

The DCFG_CC_SOCU_PIN setting specifies if the debug authentication process can modify the access rights that DCFG_CC_SOCU_DFLT defines:

- If the bit is clear, access rights to the corresponding debug domain are determined via the debug authentication process.
- If the bit is set, access rights (as defined through DCFG_CC_SOCU_DFLT) cannot be altered.

DCFG_CC_SOCU_NS_DFLT and DCFG_CC_SOCU_NS_PIN further restrict the debug access in the field. These fields are in CFPA. They translate to:

- $\text{SOCU\_PIN}[n] = 1, \text{SOCU\_DFLT}[n] = 0$: Offers the highest level of restriction: Debug access for feature n is always disabled.
- $\text{SOCU\_PIN}[n] = 0, \text{SOCU\_DFLT}[n] = 0$: This setting allows the debug authentication process to define access rights.
- $\text{SOCU\_PIN}[n] = 0, \text{SOCU\_DFLT}[n] = 1$: Illegal setting, the part may lock up if this setting is selected.
- $\text{SOCU\_PIN}[n] = 1, \text{SOCU\_DFLT}[n] = 1$: This setting is the lowest security level: Debug access for feature n is always enabled.

The flexibility in defining configuration allows module makers to implement a tiered protection approach. Tier-1 developers can develop secure code and define access rights to this module using DCFG_CC_SOCU_DFLT and DCFG_CC_SOCU_PIN. Configuration can be such that debug access to a secure module is blocked but a nonsecure debug is allowed. When ready, the tier-1 developer can release the part to a tier-2 developer, but block debug access to the secure module. The tier-2 developer can develop the nonsecure module and extend the access rights configuration to that module using DCFG_CC_SOCU_NS_DFLT and DCFG_CC_SOCU_NS_PIN. These fields are in CFPA. The following seven credentials are defined in DCFG_CC_SOCU:

- Bit 0: NIDEN - nonsecure noninvasive debug.
- Bit 1: DBGEN - nonsecure invasive debug.
- Bit 2: SPNIDEN - secure noninvasive debug.
- Bit 3: SPIDEN - enable secure invasive debug.

- Bit 4: TAPEN - enable test access point (TAP) controller.
- Bit 6: ISPCMDEN - enable boot flow commands.
 - Enables command to boot from specific ISP mode.
- Bit 7: PMCMDEN - enable permanent modification commands.
 - Erases keys stored in PFR.
 - Set FA flag field in PFR to put part in field return analysis mode.
 - Enables the command to mass-erase of user flash area.
- ROM implements a 4-byte field in PFR. The upper 2 bytes from the factory area (CMPA.VENDOR_USAGE.VU_UHW) and the lower 2 bytes from the in-field area (CFPA.VENDOR_USAGE.VU_LHW).
- The DCFG_UUID field is derived the same way as the fields in DCFG_CC_SOCU.
 - If DCFG_UUID is set, the ROM must accept debug credentials (DC) which only match the UUID of the device.

43.6.3 Debug authentication challenge (DAC)

The debug authentication protocol begins with a debug authentication challenge (DAC) message that the device issues to the debugger. The total message size is 104 bytes. [Figure 127](#) shows the named elements of this message.

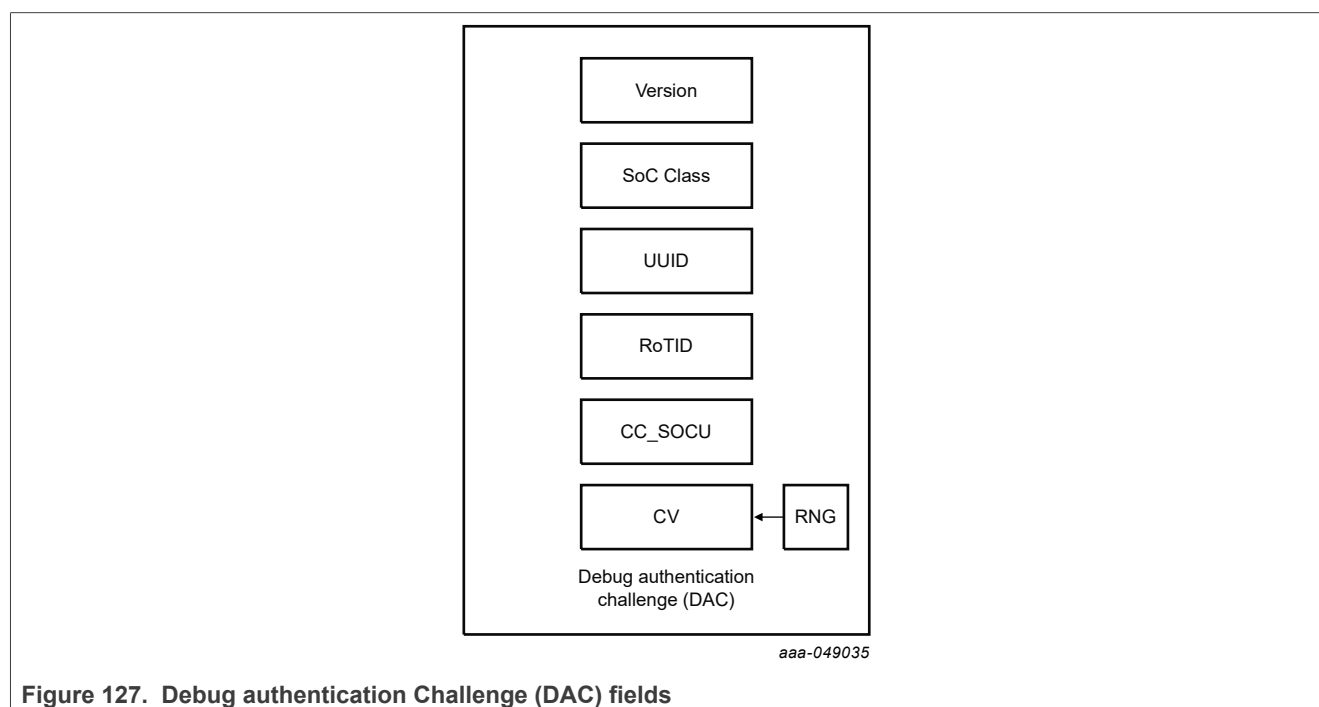


Figure 127. Debug authentication Challenge (DAC) fields

1. VER, the protocol version that the device supports:
 - Based on the DCFG_RSA4K flag in PFR, ROM sets the version field to 1.0 or 1.1.
 - 4 bytes.
2. SoCC, the SoC class of the device hardware
 - 4 bytes. Value for MCX W23: 0x0001
3. UUID, the UUID that the credential must contain to make UUID effective, otherwise all zeroes to indicate that UUID matching is not in effect.
 - 16 bytes, stored in DCFG_UUID
4. RoTID, the root of trust identifier specified in device configuration
 - 36 bytes.
 - SHA256 hash of root of trust public keys hash table, stored in DCFG_RKTH.
5. CC, the credential constraints specified in device configuration
 - 12 bytes.
 - Based on DCFG_CC_SOCU and DCFG_CC_SOCU_NS.
6. CV, the challenge vector that the device generates.
 - 32-byte random value generated using the TRNG block.

43.6.4 Debug authentication response (DAR)

Before the debugger can formulate a response to the challenge generated through DAC, it must perform some checks to confirm correctness of VER, SoCC, UUID, RoTID, and CC. After the checks, it must find a matching DC.

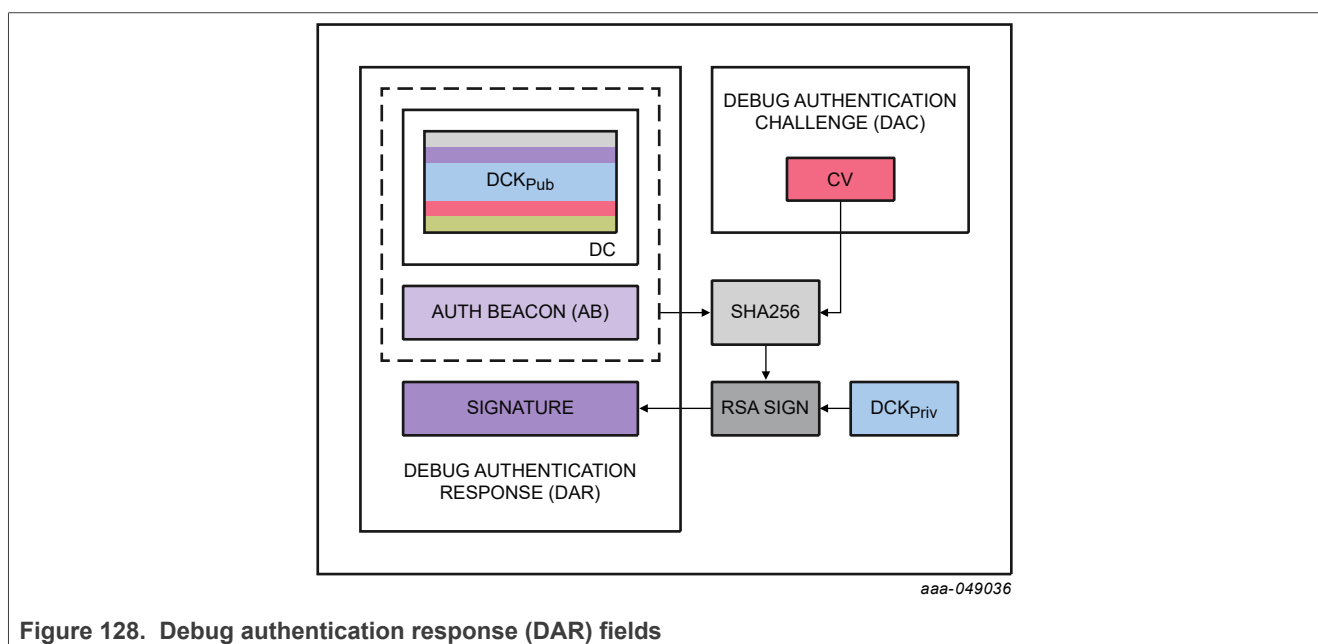


Figure 128. Debug authentication response (DAR) fields

Next, the debugger forms a debug authentication response (DAR) message, which consists of the following named elements. The total size of the DAR message is 1200 bytes (v1.0) or 2224 bytes (v1.1).

- DC, provides the credential of the debugger, RoT public key, and more, described in [Section 43.6.1](#).
 - 940 bytes (v1.0) or 1708 bytes (v1.1).
- AB, the authentication beacon that the debugger provides and signs during the authentication process.
 - 4 bytes.
- SIG, A cryptographic signature by the debugger that binds the previous two fields with the challenge vector from the DAC.
{ QUOTE SIG } (DCK_{priv}, SHA256(DAR::DC || DAR::AB || 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 bytes (v1.0) or 512 bytes (v1.1).

43.6.5 Debug port states before entering user code

Before entering the user code or the failure loop, ROM executes the following steps.

1. ROM checks if SWD_ACCESS_CPU0 is set to the enable value (0x12345678 has been written to it; reads back as 0xA).
 - If yes, ROM entry is due to non-POR reset. So, ROM copies the DEBUG_FEATURES register value to the DEBUG_FEATURES_DP register and exits to the user code/failure loop.
2. If DEBUG_AUTH_BEACON register is set to zero:
 - Check TAPEN bit in the Debug_State.
 - If set, the unlock value (0x12345678) is written to CODESECURITYPROTTEST register.
 - If not set, the disable port value (0x00000000) is written to CODESECURITYPROTTEST register.
 - Check states of NIDEN, DBGEN, SPNIDEN, SPIDEN bits in Debug_State.
 - If any of the bits is set, the unlock value (0x12345678) is written to SWD_ACCESS_CPU0 register.
 - For bits which are set, the enable value (b'10) is written in corresponding bit fields of DEBUG_FEATURES register.
 - For bits which are not-set, the disable value (b'01) is written in corresponding bit fields of DEBUG_FEATURES register.
 - If all bits are zero, the disable port value (0x00000000) is written to SWD_ACCESS_CPU0 register.
 - The DEBUG_FEATURES register value is copied to DEBUG_FEATURES_DP register.
 - The debug control registers are write-locked by writing to the DEBUG_LOCK_EN register.
 - Exits to the user code/failure loop.
3. If DEBUG_AUTH_BEACON is non-zero, ROM defers the unlocking/enabling of the debug ports to user code.
 - Check TAPEN bit in the Debug_State.
 - If set, the CODESECURITYPROTTEST register is not written.
 - If not set, the disable port value (0x00000000) is written to CODESECURITYPROTTEST register.
 - If all of the following bits are set in the Debug_State, the SWD_ACCESS_CPU0, and DEBUG_FEATURES_DP registers are not written. Otherwise, the registers are written as per the Debug_State value and write access is locked by setting the DEBUG_LOCK_EN register.
 - DFLT_NIDEN, DFLT_DBGEN, DFLT_SPNIDEN, DFLT_SPIDEN
 - In all cases, ROM writes to the DEBUG_FEATURES register, based on the Debug_State.

43.6.6 Fault analysis (FA) mode

Depending on the chip, the ROM offers an FA mode (SET_FA_MODE) command handler to enable the deletion of sensitive information (for example, keys) before handing over the device to NXP for fault analysis. ROM only allows the SET_FA_MODE command when the corresponding flag in the Debug_State is set. The FA_MODE boot sequence is activated, which ensures that:

- A new version of the customer field programmable area (CFPA) page is created.
- The ENABLE_FA_MODE word in the page is set to the value 0xC33CA55A and reset, which has the following effect:
 - The temporary key registers in hardware are cleaned
 - The PUF indexes are blocked.
 - All keys in KEYSTORE flash page are erased.
 - All debug ports are opened
 - The ROM enters while(1) loop.

If the FA mode was triggered through the mailbox command instead, the PUF is also zeroized.

43.6.7 Device processing the DAR

The device BootROM processes the DAR received from debugger. As a part of the validation step, the device:

1. Verify DC: Validate the DC version, SoCC, UUID, RoT, VU, and the DC signature.
2. Verify that the DAR has a valid signature that binds it to the CV from the DAC.

If all the steps have successfully completed, the conclusion is:

- 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 that specific 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 the authentication is successful, debug access is granted. If the authentication fails, no special response is issued. However, further debug access requests are ignored and the device enters a failure loop (infinite loop waiting for debug attach).

43.6.7.1 Successful authentication

When the debug authentication is successful, ROM executes the following steps:

1. ROM determines the final enable states of the debug ports based on the pinned state from the DCFG_CC_SOCU and the DC::CC fields.
2. ROM evaluates the port enables, using the following logic:
 - Uses the pinned states, based on the DCFG_CC_SOCU and the DCFG_CC_SOCU_NS PFR words.
 - Evaluates SOCU_PIN and SOCU_DFLT.
 - Evaluates the debug port enables for ports, which are not pinned using the authentication protocol.
 - $\text{Debug\_State} = (\text{SOCU\_PIN and SOCU\_DFLT}) | (!\text{SOCU\_PIN and DC::CC\_SOCU})$.
 - Enables debug ports for bits, which are set in the above evaluation.
3. In the debug mailbox, the handler only allows the following commands if the enable bit is set in the final evaluation of the Debug_State.
 - Only handles the 'ENTER_ISP_MODE' command if the default ISPCMDEN bit is set in the Debug_State.

- Only Handles the 'SET_FA_MODE' and the 'ERASE_FLASH' commands if the default PMCMDEN bit is set in the Debug_State.
- ROM stores the beacons in the write-lockable register, DEBUG_AUTH_BEACON (offset: 0x0000 0FC0).
 - DEBUG_AUTH_BEACON [15:0] = DAR::DC::CB[15:0]
 - DEBUG_AUTH_BEACON [31:16] = DAR::AB[15:0]
 - On receiving the EXIT_DBG_MB command, ROM exits the debug mailbox handler loop and continues the normal boot flow.

43.6.7.2 Debug port states before entering user code

Before entering into the user code or the failure loop, ROM executes the following steps:

- ROM checks if SWD_ACCESS_CPU0 is set to enable value (0x12345678).
 - If yes, ROM entry is due to non-POR reset. So, ROM copies the DBG_FEATURES register value to the DBG_FEATURES_DP register and exits to the user code/failure loop.
- If the DEBUG_AUTH_BEACON register is set to '0'
 - Check the TAPEN bit in the Debug_State
 - If set, the unlock value (0x12345678) is written to CODESECURITYPROTTEST register.
 - If not set, the disable port value (0x00000000) is written to CODESECURITYPROTTEST register.
 - Check the states of NIDEN, DBGEN, SPNIDEN, SPIDEN bits in the Debug_State.
 - If any of the bits is set, the unlock value (0x12345678) is written to SWD_ACCESS_CPU0 register.
 - For bits which are set, the enable value (b'10) is written in corresponding bit fields of DBG_FEATURES register.
 - For bits which are not set, the disable value (b'01) is written in corresponding bit fields of DBG_FEATURES register.
 - If all bits are zero, the disable port value (0x00000000) is written to SWD_ACCESS_CPU0 register.
 - The DBG_FEATURES register value is copied to the DBG_FEATURES_DP register.
 - The debug control registers are write-locked by writing to DBG_LOCKEN register.
 - Exits to the user code/failure loop.
- If DBG_AUTH_SCRATCH is non-zero, the unlocking/enabling of the debug ports to the user code is deferred.
 - Check the DFLT_TAPEN bit in the Debug_State.
 - If set, the CODESECURITYPROTTEST register is not written.
 - If not set, the disable port value (0x00000000) is written to the CODESECURITYPROTTEST register.
 - If all of the following bits are set in the Debug_State, the CODESECURITYPROTCPU0 and the DBG_FEATURES_DP registers are not written. Otherwise, the registers are written as per the Debug_State value and write access is locked by setting the DBG_LOCKEN register.
 - DFLT_NIDEN, DFLT_DBGEN, DFLT_SPNIDEN, DFLT_SPIDEN.
 - In all cases, ROM writes to the DBG_FEATURES register, based on the Debug_State.

43.6.8 Glossary

Table 221. Glossary

Term	Description
root of trust (RoT)	Vendor-owned key pair that authorizes data assets via cryptographic signatures. The public part of the key is typically preconfigured in products so that data from untrusted sources can be cryptographically verified.
RoT public key (RoTpub)	The vendor public key that the device uses to verify the signature of this DC (The corresponding private key was used to sign the DC).

Table 221. Glossary...continued

Term	Description
RoT identifier (RoTID)	RoTID allows the debugger to infer which RoT public keys are acceptable to the device. If the debugger cannot or does not provide such a credential, the authentication process fails.
RoT metadata (RoT META)	The device requires the RoT META to corroborate the RoTID sent in the DAC, the RoTK fields in this DC, and any additional RoT state that is not stored within the device. Allows different RoT identifications, management, and revocation solutions to be handled.
SoC class (SoCC)	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 the credentials. If the granularity of debug control warrants it, a class can contain just a single revision of a single SoC model.
debug credential key (DCK)	A user-owned key pair. The public part of the key is associated with a DC. The user holds the private part. It is used to produce signatures during authentication.
debug credential (DC)	A user public key and associated attributes, bound together and signed by a RoT, serves as an identity.
credential constraint (CC)	In product configuration, CCs are limitations on the DCs that the device accepts for authentication. In DCs, CCs are vendor/RoT-authorized usages of the DC and inputs to the desired debug behavior.
vendor usage (VU)	A CC (constraint) value that is opaque to the debug authentication protocol itself, but which the vendors can leverage in product-specific ways.
SoC usage (SoCU)	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. However, the device-specific debug options can also be managed in this way.
credential beacon (CB)	A value that is passed through the authentication protocol, which the protocol does not interpret, but is instead made visible to the application that is debugged. A credential beacon is associated with a DC and so is vendor/RoT-signed.
authentication beacon (AB)	A value that is passed through the authentication protocol, which the protocol does not interpret, but is instead made visible to the application that is debugged. The debugger provides and signs an authentication beacon during the authentication process.
device config (DCFG*)	Refers to device configuration settings stored in PFR.

44 2.4 GHz radio (Bluetooth Low Energy)

The 2.4 GHz radio is compatible with Bluetooth Low Energy 5.3.

44.1 Features

- The MCX W23 is compatible with the Bluetooth Low Energy 5.3 specification:
 - Bluetooth Low Energy 5.3 controller subsystem (QDID 200592)
 - Bluetooth Low Energy 5.3 host subsystem (QDID 226395)
 - Includes a 48-bit unique bluetooth device address
 - Up to 4 simultaneous connections supported
- The MCX W23 supports the following Bluetooth Low Energy features:
 - Device privacy and network privacy modes (version 5.0)
 - Advertising extension PDUs (version 5.0)
 - Anonymous device address type (version 5.0)
 - Up to 2 Mbps data rate (version 5.0)
 - Long range (version 5.0)
 - High-duty cycle, nonconnectable advertising (version 5.0)
 - Channel selection algorithm #2 (version 5.0)
 - High output power (version 5.0)
 - Advertising channel index (version 5.1)
 - Periodic advertising sync transfer (PAST) (version 5.1)
 - Supports LE power control feature (version 5.2)
- RF antenna: 50 Ω single-ended
- RF receiver characteristics:
 - Sensitivity –94 dBm in Bluetooth Low Energy 2 Mbps
 - Sensitivity –97 dBm in Bluetooth Low Energy 1 Mbps
 - Sensitivity –100 dBm in Bluetooth Low Energy 500 kbps
 - Sensitivity –102 dBm in Bluetooth Low Energy 125 kbps
 - Accurate RSSI measurement with ± 3 dB accuracy
- Flexible RF transmitter level configurability:
 - TX mode 1 (TXM1): Range from –31 dBm to +2 dBm when V_{DD_RF} exceeds 1.1 V
 - TX mode 2 (TXM2): Range from –28 dBm to +6 dBm when V_{DD_RF} exceeds 1.7 V

44.2 Bluetooth Low Energy link layer

All roles that the Bluetooth specification specifies are supported:

- Broadcaster
- Observer
- Peripheral
- Central

Further supported functionality:

- Up to 4 simultaneous connections in any role.
- Device filtering through the programmable size of the accept lists.

The Bluetooth Low Energy link layer supports the following operation modes:

- Functional mode, the Bluetooth Low Energy link layer is controlled through logical HCI commands from the Bluetooth Low Energy host stack (see [Section 44.2.1](#)).
- 'Black box' operation mode controlled over UART HCI, which can be used for Bluetooth Low Energy qualification.
 - Includes direct-test-mode configurable through UART HCI
 - Additional API functions are included to enable:
 - Transmitting in continuous wave mode
 - RSSI measurements

Note: This mode can only run in active mode and sleep power mode (low-power modes are not supported).

44.2.1 Link Layer HCI

The link-layer HCI interface is compatible with the Bluetooth Low Energy 5.3 specification. The HCI can be used as a logical API in functional mode and over a UART interface when using the 'black box' mode of operation.

44.2.2 HCI vendor-specific commands

This section documents the vendor-specific HCI commands for the MCX W23 link layer.

44.2.2.1 HCI LE set custom event mask

Table 222. HCI LE set custom event mask command

Command	OCF	Parameters	Return value
HCI_LE_Set_Custom_Event_Mask	0x0073	LE_Custom_Event_Mask	status

Description

The HCI_LE_Set_Custom_Event_Mask command is used to control which LE events the HCI generates for the Host. If the bit in the LE_Custom_Event_Mask is set to 1, the event associated with that bit is enabled. The value with all bits sets to 0 indicates that no events are specified.

Table 223. LE custom event mask parameter (1 octet)

Parameter bit	LE subevent types
0	Reserved
1	Update sleep clock accuracy complete event
2	HCI_LE_Sync event
7:3	Reserved

Return values

Table 224. LE custom event mask return value (1 octet)

Return value	Parameter description
0x00	HCI_LE_Set_Custom_Event_Mask command succeeded.
0x01 to 0xFF	HCI_LE_Set_Custom_Event_Mask command failed.

Events generated

When the HCI_LE_Set_Custom_Event_Mask command is complete, the controller sends an HCI_Command_Complete event to the host.

44.2.2.2 HCI release version

Table 225. HCI release version command

Command	OCF	Parameters	Return value
HCI_RELEASE_VERSION	0x302	None	A string value that contains the hardware and the firmware version numbers in the following format "RTL:X.XXX-YYYXX_FIRMWARE:X.XXA- YYYXX". Where X is the release version number in major and minor YYY is the release level.

Description:

The command "HCI_RELEASE_VERSION" is used to get the current version of both firmware and hardware.

44.2.2.3 HCI set PHY TX power

Table 226. HCI set PHY TX power command

Command	OCF	Parameters	Return value
HCI_CMD_OCF_LE_WRITE_TX_PWR	0x303	TX power in dBm : signed char	Success if the PHY is properly configured with the passed value UNSUPPORTED_FEATURE_OR_PARAMETER_VALUE. The TX level passed is out of the used PHY support

Description:

The command "HCI_CMD_OCF_LE_WRITE_TX_PWR" is used to set the PHY TX power for all Bluetooth Low Energy activities. The TX power level, which is set using this command, is restored to the default TX power level after HCI_RESET is received. The TX power level received in this command is used for the HCI_CMD_OCF_LE_TRANSMITTER_TEST command, except for the V4 command. For the command HCI_CMD_OCF_LE_TRANSMITTER_TEST, it uses the TX power based on the TX_Power_Lvl parameter provided with the command.

44.2.2.4 HCI set BD address

Table 227. HCI set BD address command

Command	OCF	Parameters	Return value
HCI_CMD_OCF_LE_SET_BD_ADDRES	0x304	BD_ADDRESS	Success if the address is given properly. if no address is provided, INVALID_HCI_COMMAND_PARAMETERS.

Description

The command "HCI_CMD_OCF_LE_SET_BD_ADDRESS" is used to set the BD address that is used for all Bluetooth Low Energy activities. It must be set before creating any Bluetooth Low Energy activity.

Table 228. BD Address parameter (6 octets)

Value	Parameter description
0XXXXXXXXXXXXX	BD address used for different Bluetooth Low Energy activities. Range: 0 to 0xFFFFFFFFFFFF

44.2.2.5 HCI periodic PHY calibration

Table 229. HCI periodic PHY calibration command

Command	OCF	Parameters	Return value
HCI_CMD_OCF_LE_SET_PHY_CALIBRATION_EVENT_PARAMS	0x305	PHY calibration event period: unsigned int PHY calibration event count: unsigned int	None

Description

The command “HCI_CMD_OCF_LE_SET_PHY_CALIBRATION_EVENT_PARAMS” is used to control the periodic PHY calibration activity. This command takes two parameters:

- PHY calibration event period:
Used to control the periodicity of the execution of the PHY calibration event. It is a number of the “PHY_CALIBRATION_PERIOD_STEP”, which is 1 second. Its default value is 10 (executing the PHY calibration event every 10 seconds). 0 means that the periodic calibration is disabled.
- PHY calibration event count:
Used to control the number of PHY calibration events that must be executed. Its default value is 0, which indicates that the PHY calibration event is executed an infinite number of times.

44.2.2.6 HCI set deep sleep mode enable

Table 230. HCI set deep sleep mode enable command

Command	OCF	Parameters	Return value
HCI_CMD_OCF_LE_SET_DEEP_SLEEP_MODE_ENABLE	0x306	DEEP_SLEEP_ENABLE	None

Description

The command “HCI_CMD_OCF_LE_SET_DEEP_SLEEP_MODE_ENABLE” is used to control HW deep sleep mode. The application layer is responsible for entering and leaving the deep sleep state at the correct times. If the Deep Sleep state is enabled while a new Bluetooth Low Energy event is about to execute, this event is missed.

DEEP_SLEEP_ENABLE = 1; enable deep sleep.

DEEP_SLEEP_ENABLE = 0; disable deep sleep.

Other values are reserved for future use.

44.2.2.7 HCI get remaining time for next event

Table 231. HCI get remaining time for next event command

Command	OCF	Parameters	Return value
HCI_CMD_OCF_LE_GET_REMAINING_TIME_FOR_NEXT_EVENT	0x307	None	REMAINING_TIME

Description

The command “HCI_CMD_OCF_LE_GET_REMAINING_TIME_FOR_NEXT_EVENT” is used to get the time remaining for the next event in microseconds.

REMAINING_TIME = 0; If an event is already executed on air.

REMAINING_TIME = 0xFFFFFFFF; If no events are registered in the system.

44.2.2.8 HCI set event setup time value

Table 232. HCI set event setup time value command

Command	OCF	Parameters	Return value
HCI_CMD_OCF_LE_SET_SCHEDULER_SETUP_TIME	0x308	SETUP_TIME	None

Description

The command “HCI_CMD_OCF_LE_SET_SCHEDULER_SETUP_TIME” is used to set the setup time (in μ s) required for any event. To avoid missing events if it is too small or consume power if it is big, this value is chosen carefully. The firmware does not perform any checks on the given value because its limits are hardware-dependent.

44.2.2.9 HCI select sleep clock source

Table 233. HCI select sleep clock source command

Command	OCF	Parameters	Return value
HCI_CMD_OCF_LE_SELECT_SLEEP_CLOCK_SOURCE	0x30A	slp_clk_src	slp_clk_freq_value

Description

The command “HCI_CMD_OCF_LE_SELECT_SLEEP_CLOCK_SOURCE” is used to select the sleep clock source that is used for the sleep timer. This command must be called after the HCI_RESET command as HCI_RESET returns the clock source to its default configuration.

Command parameters

Table 234. Sleep clock source parameter (1 octet)

Value	Parameter description
0x00	A 32.000 kHz crystal oscillator is used to derive the sleep clock (no calibration is required)
0x01	An RCO clock in the range of 12 to 52 kHz is used to derive the sleep clock. Periodic calibration is required.
0x02	The 32.768 kHz RTC oscillator is used to derive the sleep clock (no calibration is required).

Return value

Table 235. Sleep clock frequency (2 octets)

Value	Parameter description
12000 to 52000	If the RCO option is selected, the nominal RCO frequency value in Hz as the controller measures it. Or, if the crystal oscillator option is selected, the crystal oscillator frequency value in Hz.

44.2.2.10 HCI set connection TX power command

Table 236. HCI set connection TX power command

Command	OCF	Parameters	Return value
HCI_CMD_OCF_LE_SET_CONNECTION_TX_POWER_LEVEL	0x30F	Connection_Handle, PHY, TX_Power_Level	status, Current_TX_Power_Level

Description

The HCI_CMD_OCF_LE_SET_CONNECTION_TX_POWER_LEVEL command is used to set the current transmit power level of the local controller on the ACL connection that the Connection_Handle parameter identifies. And on any associated physical links, such as isochronous physical links and the PHY that the PHY parameter indicates. If the local controller does not support the requested TX_Power_Level value, the controller uses the nearest transmit power level that is greater than the requested one. Also, if the TX_Power_Level value is greater than the maximum supported power level, the current transmit power level is changed to the maximum level that the controller supports. The Current_TX_Power_Level return parameter represents the actual current transmit power level that the local controller selects.

Command parameters

Table 237. Connection_Handle (2 octets)

Value	Parameter description
0xFFFF	Connection_Handle Range: 0x0000-0xFFFF

Table 238. PHY (1 octet)

Value	Parameter description
0x01	LE 1M PHY
0x02	LE 2M PHY
0x03	LE coded PHY with S = 8 data coding
0x04	LE Coded PHY with S = 2 data coding

Table 239. Transmit_Power_Level (1 octet)

Value	Parameter description
0xFF	Transmit power level Units: dBm

Return parameters:

Table 240. Status (1 octet)

Value	Parameter description
0x00	HCI_CMD_OCF_LE_SET_CONNECTION_TX_POWER_LEVEL command succeeded.
0x01 - 0xFF	HCI_CMD_OCF_LE_SET_CONNECTION_TX_POWER_LEVEL command failed. With the controller error codes as specified in the command description.

Table 241. Current_TX_Power_Level (1 octet)

Value	Parameter description
0xXX	Current TX power level that the local controller units select: dBm.

44.2.2.11 HCI set RSSI golden range command

Table 242. HCI set RSSI golden range command

Command	OCF	Parameters	Return value
HCI_CMD_OCF_SET_RSSI_GOLDEN_RANGE	0x0312	Golden_Range_Upper_Limit; Golden_Range_Lower_Limit	status

Description

The HCI_CMD_OCF_SET_RSSI_GOLDEN_RANGE command is used to select golden range upper and lower limits values, other than the values that the controller uses. When the HCI_CMD_OCF_SET_RSSI_GOLDEN_RANGE command is complete, an HCI_Command_Complete event is generated.

Command parameters

Table 243. Golden_Range_Upper_Limit (1 octet)

Value	Parameter description
0xXX	Receiver golden range upper limit. Type: signed integer

Table 244. Golden_Range_Lower_Limit (1 octet)

Value	Parameter description
0xXX	Receiver golden range lower limit. Type: signed integer

Return parameters

Table 245. Status (1 octet)

Value	Parameter description
0x00	HCI_CMD_OCF_SET_RSSI_GOLDEN_RANGE command succeeded.
0x01–0xFF	HCI_CMD_OCF_SET_RSSI_GOLDEN_RANGE command failed.

44.2.2.12 HCI select TX power mode command

Table 246. HCI select TX power mode command

Command	OCF	Parameters	Return value
HCI_CMD_OCF_LE_SELECT_TX_POWER_MODE	0x0314	TX_Power_Mode	status

Description

The HCI_CMD_OCF_LE_SELECT_TX_POWER_MODE command is used to select the TX_Power mode of the controller. TX_Power mode is selected during FW initialization and cannot be changed during the runtime. If this command is issued during runtime, the LL returns the "COMMAND_DISALLOWED" (0x0C) error code. If the "TX_Power_Mode" parameter does not identify a TX_Power mode that the selected PHY supports, the LL returns the "INVALID_HCI_COMMAND_PARAMETERS" (0x12) error code and uses the 2 dBm power mode. The "TX_Power_Mode" parameter is used to identify which TX_Power table ID must be used. By default, the 2 dBm power mode is used. When the HCI_CMD_OCF_LE_SELECT_TX_POWER_MODE command is complete, an HCI_Command_Complete event is generated.

Command parameters

Table 247. TX_Power_Mode (1 octet)

Value	Parameter description
0x00	2_dBm TX power mode
0x01	6_dBm TX power mode
0x02	10_dBm TX power mode

Return parameters

Table 248. Status (1 octet)

Value	Parameter description
0x00	HCI_CMD_OCF_LE_SELECT_TX_POWER_MODE command succeeded.
0x01–0xFF	HCI_CMD_OCF_LE_SELECT_TX_POWER_MODE command failed. With controller error codes as specified in the command description.

44.2.2.13 HCI curb sleep state

Table 249. HCI curb sleep state command

Command	OCF	Parameters	Return value
HCI_CMD_OCF_CURB_SLEEP_STATE	0x0315	enable	status

Description

The HCI_CMD_OCF_CURB_SLEEP_STATE command is used to prevent that the link layer and the PHY enter the sleep state. If this command is issued while the device is in sleep mode, it leaves the sleep state and switches to active mode.

Note: This command can only be used when the device is not in deep sleep mode.

Command parameters

Table 250. Enable (1 octet)

Value	Parameter description
0x00	Disables preventing that the link layer and PHY from enter deep sleep.
0x01	Enables preventing that the link layer and PHY from enter deep sleep.

Return parameters

Table 251. Status (1 octet)

Value	Parameter description
0x00	HCI_CMD_OCF_CURB_SLEEP_STATE command succeeded.
0x01–0xFF	HCI_CMD_OCF_CURB_SLEEP_STATE command failed. The "enable" parameter has an invalid value or the device is in deep sleep mode.

44.2.2.14 HCI locally terminate ACL connection

Table 252. HCI locally terminate ACL connection command

Command	OCF	Parameters	Return value
HCI_CMD_OCF_ACL_IMMEDIATE_DISCONNECT	0x0332	Connection_handler	status

Description

Asynchronously terminate the existing ACL connection locally without sending LL_TERMINATE_IND to the peer device. The Connection_handle is the connection handle of the running ACL connection that the host targets to terminate. If the Connection_Handle is greater than 0x0EFF, the controller returns the error code Invalid HCI Command Parameters (0x12). If the Connection_Handle parameter does not identify a current ACL connection, the controller returns the error code Unknown Connection Identifier (0x02).

Events generated (unless masked away): When the controller receives the HCI_CMD_OCF_ACL_IMMEDIATE_DISCONNECT command, it sends the HCI_Command_Status event to the host. A disconnection complete event is generated after the current ACL connection is terminated successfully with status "Connection Terminated by Local Host" (0x16).

Command parameters

Table 253. Connection_Handle (2 octets)

Value	Parameter description
0xFFFF	Connection_Handle Range: 0x0000-0x0EFF

Return parameters:

Table 254. Status (1 octet)

Value	Parameter description
0x00	Command succeeded
0x01–0xFF	HCI_CMD_OCF_ACL_IMMEDIATE_DISCONNECT command failed. With controller error codes as specified in the command description.

44.2.3 HCI external custom commands

To register a Bluetooth Low Energy external custom HCI commands callback function, the `hci_rgstr_ble_external_custom_cbk()` function must be called. It has the following definition:

Table 255. `hci_rgstr_ble_external_custom_cbk()` function

Prototype	<code>uint8_t hci_rgstr_ble_external_custom_cbk(ble_ext_custm_cb_t ext_custm_cbk)</code>
Parameters	<code>ext_custm_cbk</code>

Table 255. hci_rgstr_ble_external_custom_cbk() function...continued

Return value	True if registration succeeds; False otherwise

The callback function passed to hci_rgstr_ble_external_custom_cbk() has the following definition:

Table 256. Callback function

Callback type	ble_ext_custm_cb_t		
Parameter	uint16_t ocf	input	Unresolved Bluetooth Low Energy HCI custom OCF.
	uint8_t *pckt_p		Pointer to the received HCI command that the callback function must parse.
	uint8_t *evnt_pckt_p	output	Pointer to the parameters section of the event reported to the HCI in response to the received Bluetooth Low Energy HCI custom OCF.
	uint8_t *params_length		Length of the parameters added to the response event in bytes.
	hci_return_command_type *return_command_type		Type of the event that must be returned (a command complete event or a command status event).
Return value	command status		

When a custom Bluetooth Low Energy HCI command is received, and the firmware is unable to resolve it:

1. The firmware checks if an external custom HCI command callback function is registered. If it is found, the unresolved OCF is passed along with the HCI command received to the registered callback function.
2. The firmware expects the callback function to return with the command status as a return value and expects the output function parameters to be set according to the previously mentioned definition.
3. Depending on the return_command_type, the HCI reports a command complete event or a command status event.

An example of an external custom command can be found in the hci_black_box application.

44.2.4 Bluetooth Low Energy power management

This section documents the usage of the low-power features of the Bluetooth Low Energy radio. For a more general overview of the Power Management refer to [Section 13](#)

The radio alternates periods of activity with periods of inactivity.

Radio activity results in current peaks.

During inactive periods, the application can plan activities that are sensitive to these current peaks or put the system in a low power mode.

In active periods, the application must ensure the system is and remains awake.

The application can read the remaining radio idle time to see whether other activities fit or to calculate the wakeup time. If the radio isn't idle or the idle time is too short, a callback can be registered that is invoked when the radio again turns idle. The application then reevaluates the idle time.

The functions in the ble_controller.h used for this are:

```

/*! *****
 * \brief Check whether there are pending software tasks, i.e. pending events that
 *        require processing
 *
 * \return 0: Not idle, 1: idle.
 *
******/
```

```

***** */
int BLEController_EmngrIsIdle(void);

/*! *****
 * \brief Get the remaining timer for the next event. Safe access to internal timer
 * structures. Potentially slower and less predictable call duration.
 * \param remainingTimeUs: [out] the value of remaining time for the next event in us
 * \return blec_result_t
 ***** */
blec_result_t BLEController_GetRemainingTimeForNextEvent(uint32_t* remainingTimeUs);

/*! *****
 * \brief Get the remaining timer for the next event. Lock free but unsafe access to
 * internal timer structures. Faster and more predictable call duration.
 * \param remainingTimeUs: [out] the value of remaining time for the next event in us
 * \return blec_result_t
 ***** */
blec_result_t BLEController_GetRemainingTimeForNextEventUnsafe(uint32_t* remainingTimeUs);

typedef void (*blecIdleCallback_t)(void);

/*! *****
 * \brief Register function to be called when there are no more events to process
 *
 * \param callback : [in] callback to execute when the Controller is Idle
 ***** */
void BLEController_RegisterIdleCallback(blecIdleCallback_t callback);

```

44.2.5 TX power control

Transmit power can be controlled using the following functions in `ble_controller.h`:

```

/*! *****
 * \brief Set the transmit power level of the Controller for all BLE events,
 * exceptions are:
 * - Extended advertising events: Set by the host as a parameter of
 *   HCI_LE_Set_Extended_Advertising_Param (OCF=0x0036)
 *
 * If the requested level_dbm value is not supported by the controller but is
 * between the supported range, the controller uses the nearest transmit power level
 * that is greater than the requested one.
 *
 * \param level_dbm : [in] Desired Tx Power Level in dbm
 *
 * \return blec_result_t
 ***** */
blec_result_t BLEController_SetTxPowerDbm(int8_t level_dbm);

/*! *****
 * \brief Set the initial transmit power level of the Controller Connection
 * events.
 *
 * If the requested level_dbm value is not supported by the controller but is
 * between the supported range, the controller uses the nearest transmit power level
 * that is greater than the requested one.
 *
 * \param level_dbm : [in] Desired Tx Power Level in dbm
 *
 * \return blec_result_t
 ***** */
blec_result_t BLEController_SetConnectionInitialTxPowerDbm(int8_t level_dbm);

/*! *****
 * \brief Initialize the Controller
 *
 * \param callback : [in] callback HCI Host Receive Callback
 *
 * \param requested_max_power_dBm : [in] Maximum transmit power in dbm that the
 * Controller needs to support. The Controller will choose the power table
 * that both:
 * - supports at least up to the given value
 * - optimizes power use
 *
 * If the requested value cannot be configured in the Controller then the
 * Controller will choose the power table which maximum value is the
 * nearest higher value, if it can.
 *
 * If the requested value is higher than the maximum supported value then

```

```

*           the Controller will select the power table which maximum value is the
*           highest.
*
* \param selected_power_range : [out] min and max power values of the power table
*           which was selected by the controller based on the input.
*
*
* \return bleResult_t
*****
blec_result_t BLEController_Init(blecHostHciRecvCallback_t callback,
int8_t requested_max_power_dBm,
power_range_dBm_t *selected_power_range);

```

If a given value for `requested_max_power_dBm` is below 3 dBm, the link layer operates in TXM1 mode. This mode has a range of -31 dBm to +2 dBm. It uses the following steps:

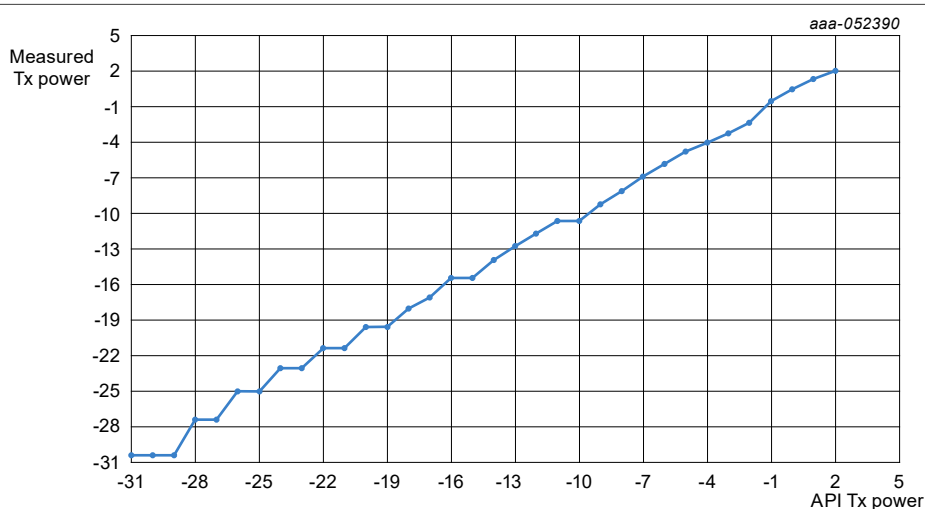


Figure 129. TX power steps in TXM1 mode

If a given value for `requested_max_power_dBm` is between 3 dBm and 6 dBm, the link layer operates in TXM2 mode. This mode has a range of -28 dBm to 6 dBm. It uses the following steps:

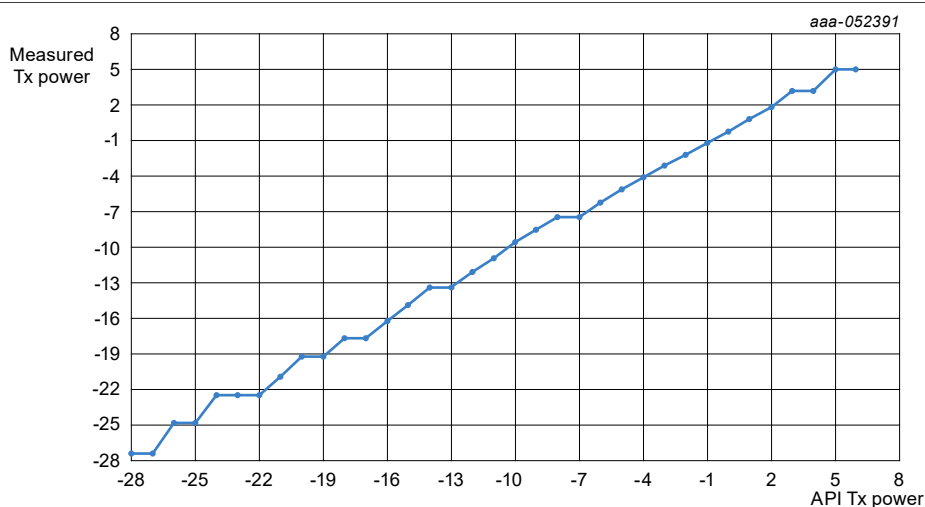


Figure 130. TX power steps in TXM2 mode

44.2.6 Continuous wave

The Bluetooth Low Energy link layer can be controlled to generate or receive a continuous wave for testing purposes.

Two flavors are implemented for the generation:

- Continuously transmit a raw carrier on a selected Bluetooth Low Energy channel.
- Continuously transmit a modulated payload on a selected Bluetooth Low Energy channel. The modulation and the PHY to use are also selectable in this case.

Continuous wave can be controlled using the following functions in `ble_controller.h`:

```
typedef enum
{
    kBLEC_CHANNEL_CW_0      = 0U,
    kBLEC_CHANNEL_CW_1      = 1U,
    kBLEC_CHANNEL_CW_2      = 2U,
    kBLEC_CHANNEL_CW_3      = 3U,
    kBLEC_CHANNEL_CW_4      = 4U,
    kBLEC_CHANNEL_CW_5      = 5U,
    kBLEC_CHANNEL_CW_6      = 6U,
    kBLEC_CHANNEL_CW_7      = 7U,
    kBLEC_CHANNEL_CW_8      = 8U,
    kBLEC_CHANNEL_CW_9      = 9U,
    kBLEC_CHANNEL_CW_10     = 10U,
    kBLEC_CHANNEL_CW_11     = 11U,
    kBLEC_CHANNEL_CW_12     = 12U,
    kBLEC_CHANNEL_CW_13     = 13U,
    kBLEC_CHANNEL_CW_14     = 14U,
    kBLEC_CHANNEL_CW_15     = 15U,
    kBLEC_CHANNEL_CW_16     = 16U,
    kBLEC_CHANNEL_CW_17     = 17U,
    kBLEC_CHANNEL_CW_18     = 18U,
    kBLEC_CHANNEL_CW_19     = 19U,
    kBLEC_CHANNEL_CW_20     = 20U,
    kBLEC_CHANNEL_CW_21     = 21U,
    kBLEC_CHANNEL_CW_22     = 22U,
    kBLEC_CHANNEL_CW_23     = 23U,
    kBLEC_CHANNEL_CW_24     = 24U,
    kBLEC_CHANNEL_CW_25     = 25U,
    kBLEC_CHANNEL_CW_26     = 26U,
    kBLEC_CHANNEL_CW_27     = 27U,
    kBLEC_CHANNEL_CW_28     = 28U,
    kBLEC_CHANNEL_CW_29     = 29U,
    kBLEC_CHANNEL_CW_30     = 30U,
    kBLEC_CHANNEL_CW_31     = 31U,
    kBLEC_CHANNEL_CW_32     = 32U,
    kBLEC_CHANNEL_CW_33     = 33U,
    kBLEC_CHANNEL_CW_34     = 34U,
    kBLEC_CHANNEL_CW_35     = 35U,
    kBLEC_CHANNEL_CW_36     = 36U,
    kBLEC_CHANNEL_CW_37     = 37U,
    kBLEC_CHANNEL_CW_38     = 38U,
    kBLEC_CHANNEL_CW_39     = 39U,
    kBLEC_CHANNEL_CW_MAX    = kBLEC_CHANNEL_CW_39,
} channel_selection_cw_t;

typedef enum
{
    kBLEC_PAYLOAD_TYPE_CW_PRBS9      = 0U,
    kBLEC_PAYLOAD_TYPE_CW_11110000   = 1U,
    kBLEC_PAYLOAD_TYPE_CW_10101010   = 2U,
    kBLEC_PAYLOAD_TYPE_CW_PRBS15      = 3U,
    kBLEC_PAYLOAD_TYPE_CW_11111111    = 4U,
    kBLEC_PAYLOAD_TYPE_CW_00000000     = 5U,
    kBLEC_PAYLOAD_TYPE_CW_00001111     = 6U,
    kBLEC_PAYLOAD_TYPE_CW_01010101     = 7U,
    kBLEC_PAYLOAD_TYPE_CW_MAX,
} payload_type_t;

typedef enum {
    kBLEC_PHY_1M      = 0x01,
    kBLEC_PHY_2M      = 0x02,
    kBLEC_PHY_CODED_S8 = 0x03, /* S=8, where 8 symbols represent 1 bit */
    kBLEC_PHY_CODED   = 0x04, /* S=2, where 2 symbols represent 1 bit */
} tx_rx_phy_t;

/*! *****
 * \brief Start continuous wave transmission
```

```

*
* \param channel      : [in] Channel where the wave will be transmitted
*
*****
blec_result_t BLEController_ContinuousWave_StartTx(channel_selection_cw_t channel);

/*! *****
* \brief Start modulated continuous wave transmission
*
* \param channel      : [in] Channel where the wave will be transmitted
* \param payload_type : [in] Payload type
* \param phy          : [in] Modulation/Phy to use
*
***** */
blec_result_t BLEController_ContinuousWave_StartTxModulated(channel_selection_cw_t channel,
payload_type_t payload_type, tx_rx_phy_t phy);

/*! *****
* \brief Start continuous wave reception
***** */
blec_result_t BLEController_ContinuousWave_StartRx(channel_selection_cw_t channel);

/*! *****
* \brief Stop Continuous Wave operation
***** */
blec_result_t BLEController_ContinuousWave_Stop(void);

/*! *****
* \brief Used to return the RSSI measured during a continuous wave reception
*
* \return rssi as signed 16-bit value in 0.25 dBm unity steps.
*
***** */
int16_t BLEController_ContinuousWave_ReadRSSI(void);

```

After starting a continuous wave operation, `BLEController_ContinuousWave_Stop` must be called before a new one is started. The only exception is `BLEController_ContinuousWave_ReadRSSI`, which is designed to be called while `BLEController_ContinuousWave_StartRx` is running.

44.3 Bluetooth Low Energy host

The MCX W23 comes with an integrated Bluetooth Low Energy host stack library, compatible with Bluetooth Low Energy version 5.3. The Bluetooth Low Energy host stack is a port of the KW45 Bluetooth Low Energy host functionality, which is documented in the "KW45 BLE Host stack documentation" ([Ref 4](#)).

45 Register description

This section gives an overview of all registers in the MCX W23 devices. It can be split in the following categories:

Table 257. Register categories

Title	Section
Embedded Flash	Section 45.1
SRAM	SRAM
External Flash	Section 45.3
APB peripherals	Section 45.4
AHB peripherals	Section 45.5
Arm NVIC and TrustZone	Section 45.6

45.1 Flash

The main flash area is located at base address 0000_0000h. Depending on the selected device, the size of the main flash array can be different. In addition to the main flash array, every device also contains protected flash regions (PFR). The sections below give an overview of the bits in these PFR.

Protected flash region	Base Address	Reference
CFPA scratch	000F_E000h	Section 45.1.1
CFPA ping	0010_0000h	Section 45.1.1
CFPA pong	0010_2000h	Section 45.1.1
CMPA	0010_4000h	Section 45.1.2
OEM key store	0010_5000h	Section 45.1.3
NXP key store	0010_7800h	Section 45.1.4
NMPA	0010_8000h	Section 45.1.6

45.1.1 Customer field programmable area (CFPA)

The PFR API can be used to program the CFPA scratch area. When rebooting, the bootloader checks the version of the data that is in the CFPA scratch page. When there is a more recent version in the scratch page, copies it to either the CFPA ping or CFPA pong page. The ROM uses the CFPA ping or the CFPA pong page with the higher version number as the active page.

45.1.1.1 CFPA memory map

[Table 258](#) gives an overview of all fields in the CFPA page. The bit fields and functionality are the same for the CFPA scratch and the CFPA ping and pong pages. The following base addresses are used for the different pages:

- CFPA SCRATCH base address: 000F_E000h
- CFPA PING base address: 0010_0000h
- CFPA_PONG base address: 0010_2000h

Table 258. CFPA memory map overview

Name	Access	Offset	Description	Width (bits)	Reset value	Section
HEADER	R/W	0h	CFPA header	32	0000_0000h	Section 45.1.1.2.1
VERSION	R/W	4h	CFPA version	32	0000_0000h	Section 45.1.1.2.2
S_FW_Version	R/W	8h	secure firmware version	32	0000_0000h	Section 45.1.1.2.3
NS_FW_Version	R/W	Ch	non-secure firmware version	32	0000_0000h	Section 45.1.1.2.4
IMAGE_KEY_REVOKE	R/W	10h	image key revocation ID	32	0000_0000h	Section 45.1.1.2.5
ROTKH_REVOKE	R/W	18h	individual root keys revocation control	32	0000_0000h	Section 45.1.1.2.6
VENDOR_USAGE	R/W	1Ch	reserved for vendor use	32	0000_0000h	Section 45.1.1.2.7
DCFG_CC_SOCU_NS_PIN	R/W	20h	security policy for debug access control pin configuration	32	0000_0000h	Section 45.1.1.2.8

Table 258. CFPA memory map overview...continued

Name	Access	Offset	Description	Width (bits)	Reset value	Section
DCFG_CC_SOCU_NS_DFLT	R/W	24h	security policy for debug access control default configuration	32	0000_0000h	Section 45.1.1.2.9
ENABLE_FA_MODE	R/W	28h	enable FA mode	32	0000_0000h	Section 45.1.1.2.10
CMPA_PROG_IN_PROGRESS	R/W	2Ch	CMPA page programming ongoing	32	0000_0000h	Section 45.1.1.2.11
CUSTOMER_DEFINED0 - CUSTOMER_DEFIND87	R/W	80h - 1DCh	customer-defined CFPA fields	32	0000_0000h	Section 45.1.1.2.12
SHA256_DIGEST0 - SHA256_DIGEST7	R/W	1E0h - 1FCh	SHA256 digest for CFPA	32	0000-0000h	Section 45.1.1.2.13

45.1.1.2 Register descriptions

45.1.1.2.1 CFPA header register (HEADER)

Table 259. CFPA header register - offset: 0h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	32-bit CFPA header field	0

45.1.1.2.2 CFPA version (VERSION)

Table 260. CFPA version register - offset: 4h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	version number of the CFPA page Programmed as a monotonic counter.	0

45.1.1.2.3 Secure firmware version (S_FW_Version)

Table 261. Secure firmware version register - offset: 8h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	version number of secure firmware Programmed as a monotonic counter.	0

45.1.1.2.4 Non-secure firmware version (NS_FW_Version)

Table 262. Non-secure firmware version register - offset: Ch

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	version number of non-secure firmware	0

Table 262. Non-secure firmware version register - offset: Ch

Bit	Name	Access	Description and value	Reset value
			Programmed as a monotonic counter.	

45.1.1.2.5 Image key revocation ID (IMAGE_KEY_REVOKE)

Table 263. Image key revocation ID register - offset: 10h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	image key revocation ID Programmed as a monotonic counter.	0

45.1.1.2.6 Individual root keys revocation control (ROTKH_REVOKE)

Table 264. Individual root keys revocation control - offset: 18h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined. Only zero must be written.	-
7:6	RoTK3_EN	R/W	RoT key 3 enable 00 invalid 01 enabled 10; 11 key revoked	0
5:4	RoTK2_EN	R/W	RoT key 2 enable 00 invalid 01 enabled 10; 11 key revoked	0
3:2	RoTK1_EN	R/W	RoT key 1 enable 00 invalid 01 enabled 10; 11 key revoked	0
1:0	RoTK0_EN	R/W	RoT key 0 enable 00 invalid 01 enabled 10; 11 key revoked	0

45.1.1.2.7 Reserved for vendor use (VENDOR_USAGE)

Table 265. Reserved for vendor use - offset: 1Ch

Bit	Name	Access	Description and value	Reset value
31:16	INVERSE_VALUE	R/W	inverse value of bits [15:0] of this register	0

Table 265. Reserved for vendor use - offset: 1Ch...continued

Bit	Name	Access	Description and value	Reset value
15:0	DBG_VENDOR_USAGE	R/W	Lower 16 bits of vendor usage field defined in DAP	0

45.1.1.2.8 Security policy for debug access control pin configuration (DCFG_CC_SOCU_NS_PIN)

Level 1 customers (secure code developer) to level-2 customers who develop non-secure code can only sell this part with TrustZone for Arm M processor family (TZ-M)

- For easy development, a level-1 customer releases the part to allow non-secure debug at all times in this scenario.
- To allow level-2 customers to further seal the part, DCFG_CC_SOCU_NS is used.
- ROM uses this word to further restrict the debug access.

Table 266. Security policy for debug access control pin configuration - offset: 20h

Bit	Name	Access	Description and value	Reset value
31:16	INVERSE_VALUE	R/W	inverse value of bits [15:0] of this register	0
15	UUID_CHECK	R/W	enforce UUID match during debug authentication	0
14:8	reserved	-	reserved Read value is undefined. Only zero must be written.	-
7	PMCM DEN	R/W	programmers mode command enable	0
			0b use DAP to enable	
			1b fixed state	
6	ISPCMDEN	R/W	ISP boot command enable	0
			0b use DAP to enable	
			1b fixed state	
5	reserved	-	reserved Read value is undefined. Only zero must be written.	-
4	TAPEN	R/W	JTAG TAP enable	0
			0b use DAP to enable	
			1b fixed state	
3	SPIDEN	R/W	secure invasive debug enable	0
			0b use DAP to enable	
			1b fixed state	
2	SPNIDEN	R/W	secure non-invasive debug enable	0
			0b use DAP to enable	
			1b fixed state	
1	DBGEN	R/W	non-secure debug enable	0
			0b use DAP to enable	
			1b fixed state	
0	NIDEN	R/W	non-secure non-invasive debug enable	0
			0b use DAP to enable	
			1b fixed state	

45.1.1.2.9 Security policy for debug access control default configuration (DCFG_CC_SOCU_NS_DFLT)

With TZ-M, level 1 customers (secure code developer) to level-2 customers who develop non-secure code can only sell this part with TZ-M.

- For easy development, a level-1 customer releases the part to allow non-secure debug at all times in this scenario.
- To allow level-2 customers to further seal the part, DCFG_CC_SOCU_NS is used.
- ROM uses this word to further restrict the debug access.

Table 267. Security policy for debug access control default configuration - offset: 24h

Bit	Name	Access	Description and value	Reset value
31:16	INVERSE_VALUE	R/W	inverse value of bits [15:0] of this register	0
15:8	reserved	-	reserved Read value is undefined. Only zero must be written.	-
7	PMCM DEN	R/W	programmers mode command fixed state	0
			0b disable	
			1b enable	
6	ISPCMDEN	R/W	ISP boot command fixed state	0
			0b disable	
			1b enable	
5	reserved	-	reserved Read value is undefined. Only zero must be written.	-
4	TAPEN	R/W	JTAG TAP fixed state	0
			0b disable	
			1b enable	
3	SPIDEN	R/W	secure invasive debug fixed state	0
			0b disable	
			1b enable	
2	SPNIDEN	R/W	secure non-invasive debug fixed state	0
			0b disable	
			1b enable	
1	DBGEN	R/W	non-secure debug fixed state	0
			0b disable	
			1b enable	
0	NIDEN	R/W	non-secure non-invasive debug fixed state	0
			0b disable	
			1b enable	

45.1.1.2.10 Enable FA mode (ENABLE_FA_MODE)

Enable FA mode. To indicate boot ROM to enter FA mode, the SET_FA_MODE command must write C33C_A55Ah to this word.

Table 268. Enable FA mode - offset: 28h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	enable FA mode When this register has the value C33C_A55Ah at the start of the boot operation, the boot ROM enters FA mode.	0

45.1.1.2.11 CMPA page programming ongoing (CMPA_PROG_IN_PROGRESS)

CMPA page programming ongoing. This field is set to 5CC5_5AA5h in the active CFPA page each time CMPA page programming is going on. It is always set to 0000_0000h in the CFPA scratch area.

Table 269. CMPA page programming ongoing - offset: 2Ch

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	CMPA page programming ongoing	0

45.1.1.2.12 Customer-defined CFPA fields (CUSTOMER_DEFINEDa)

For a = 0 to 87.

Table 270. Customer-defined CFPA fields - offset: 80h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	customer-defined CFPA fields	0

45.1.1.2.13 SHA256 digest for CFPA (SHA256_DIGESTa)

For a = 0 to 7.

Table 271. SHA256 digest for CFPA - offset: 1E0h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	256-bit SHA digest field This value is calculated over all data in the CFPA page from byte offset 000h to 1DFh	0

45.1.2 Customer manufacturing programmable area (CMPA)

The CMPA can be programmed by the customer during manufacturing of their product.

45.1.2.1 CMPA memory map

Table 272. FLASH_CMPA base address: 0010_4000h

Name	Access	Offset	Description	Width (bits)		Section
BOOT_CFG	R/W	0h	boot configuration	32	0000_0000h	Section 45.1.2.2.1
SPI_FLASH_CFG	R/W	4h	SPI flash configuration	32	0000_0000h	Section 45.1.2.2.2
CC_SOCU_PIN	R/W	8h	security policy for debug access control pin configuration	32	0000_0000h	Section 45.1.2.2.3
CC_SOCU_DFLT	R/W	Ch	security policy for debug access control default configuration	32	0000_0000h	Section 45.1.2.2.4
VENDOR_USAGE	R/W	10h	reserved for vendor usage	32	0000_0000h	Section 45.1.2.2.5
SECURE_BOOT_CFG	R/W	14h	secure boot configuration	32	0000_0000h	Section 45.1.2.2.6
PRINCE_CFG	R/W	1Ch	Prince configuration	32	0000_0000h	Section 45.1.2.2.7
BMAPCTX0	R/W	20h	bitmap control for memory context 0	32	0000_0000h	Section 45.1.2.2.8
BMAPCTX1	R/W	24h	bitmap control for memory context 1	32	0000_0000h	Section 45.1.2.2.9
BMAPCTX2	R/W	28h	bitmap control for memory context 2	32	0000_0000h	Section 45.1.2.2.10
BMAPCTX3	R/W	2Ch	bitmap control for memory context 3	32	0000_0000h	Section 45.1.2.2.11
FLASH_REMAP_SIZE	R/W	58h	flash remap size	32	0000_0000h	Section 45.1.2.2.12
FLASH_REMAP_OFFSET	R/W	5Ch	flash remap offset	32	0000_0000h	Section 45.1.2.2.13
ROTKH0 - ROTKH7	R/W	60h - 7Ch	root of trust keys table hash	32	0000_0000h	Section 45.1.2.2.14
CUSTOMER_DEFINED0 - CUSTOMER_DEFINED87	R/W	80h - 1DCh	customer defined	32	0000_0000h	Section 45.1.2.2.15
SHA256_DIGEST0 - SHA256_DIGEST7	R/W	1E0h - 1FCh	SHA256 digest for CMPA	0000_0000h		Section 45.1.2.2.16

45.1.2.2 Register descriptions

45.1.2.2.1 Boot configuration (BOOT_CFG) register

Table 273. Boot configuration - offset: 0h

Bit	Name	Access	Description and value	Reset value
31:24	BOOT_FAILURE_pin	R/W	GPIO port and pin number to use for indicating failure reason The toggle rate of the pin is used to decode the error type; [2:0] - Defines GPIO port ; [7:3] - Defines GPIO pin	0
23:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8:7	BOOT_SPEED	R/W	core clock 00b Use HW default 01b 12 MHz FRO 10b 32 MHz FRO	0
6:4	DEFAULT_ISP_MODE	R/W	default ISP mode 000b auto ISP 001b reserved 010b UART ISP 011b SPI slave ISP 100b I ² C slave ISP 111b disable ISP fall through	0
3:0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.1.2.2.2 SPI flash configuration (SPI_FLASH_CFG) register

Table 274. SPI flash configuration - offset: 4h

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved Read value is undefined, only zero must be written.	-
4:0	SPI_RECOVERY_BOOT_EN	R/W	if non-zero value is written to this field, SPI flash recovery boot is enabled	0

45.1.2.2.3 Security policy for debug access control pin configuration (CC_SOCU_PIN) register

Table 275. Security policy for debug access control pin configuration - offset: 8h

Bit	Name	Access	Description and value	Reset value
31:16	INVERSE_VALUE	R/W	inverse value of bits [15:0] of this register	0
15	UUID_CHECK	R/W	enforce UUID match during debug authentication	0
14:8	reserved	-	reserved Read value is undefined. Only zero must be written.	-
7	PMCM DEN	R/W	programmers mode command enable	0
			0b use DAP to enable	
			1b fixed state	
6	ISPCMDEN	R/W	ISP boot command enable	0
			0b use DAP to enable	
			1b fixed state	
5	reserved	-	reserved Read value is undefined. Only zero must be written.	-
4	TAPEN	R/W	JTAG TAP enable	0
			0b use DAP to enable	
			1b fixed state	
3	SPIDEN	R/W	secure invasive debug enable	0
			0b use DAP to enable	
			1b fixed state	
2	SPNIDEN	R/W	secure non-invasive debug enable	0
			0b use DAP to enable	
			1b fixed state	
1	DBGEN	R/W	non-secure debug enable	0
			0b use DAP to enable	
			1b fixed state	
0	NIDEN	R/W	non-secure non-invasive debug enable	0
			0b use DAP to enable	
			1b fixed state	

45.1.2.2.4 Security policy for debug access control default configuration (CC_SOCU_DFLT) register

Table 276. Security policy for debug access control default configuration - offset: Ch

Bit	Name	Access	Description and value	Reset value
31:16	INVERSE_VALUE	R/W	inverse value of bits [15:0] of this register	0
15:8	reserved	-	reserved Read value is undefined. Only zero must be written.	-
7	PMCM DEN	R/W	programmers mode command fixed state	0
			0b disable	
			1b enable	
6	ISPCMDEN	R/W	ISP boot command fixed state	0
			0b disable	
			1b enable	
5	reserved	-	reserved Read value is undefined. Only zero must be written.	-
4	TAPEN	R/W	JTAG TAP fixed state	0
			0b disable	
			1b enable	
3	SPIDEN	R/W	secure invasive debug fixed state	0
			0b disable	
			1b enable	
2	SPNIDEN	R/W	secure non-invasive debug fixed state	0
			0b disable	
			1b enable	
1	DBGEN	R/W	non-secure debug fixed state	0
			0b disable	
			1b enable	
0	NIDEN	R/W	non-secure non-invasive debug fixed state	0
			0b disable	
			1b enable	

45.1.2.2.5 Reserved for vendor use (VENDOR_USAGE) register

Table 277. Reserved for vendor use - offset: 10h

Bit	Name	Access	Description and value	Reset value
31:16	INVERSE_VALUE	R/W	inverse value of bits [15:0] of this register	0
15:0	reserved	-	reserved Read value is undefined. Only zero must be written.	-

45.1.2.2.6 Secure boot configuration (SECURE_BOOT_CFG) register

Table 278. Secure boot configuration - offset: 14h

Bit	Name	Access	Description and value		Reset value
31:30	SEC_BOOT_EN	R/W	secure boot enable		0
			00	plain image (internal flash with or without CRC)	
			01; 10; 11	boot signed images (internal flash, RSA signed)	
29:24	reserved	-	reserved Read value is undefined. Only zero must be written.		-
23:22	BOOT_SEED_INC_SEC_EPOCH	R/W	include security EPOCH in BOOT_SEED computation On MCX W23 security epoch consists of following fields in CFPA page: • IMAGE_KEY_REVOKE • ROTKH_REVOKE • VENDOR_USAGE • DCFG_CC_SOCU_NS_PIN • DCFG_CC_SOCU_NS_DFLT		0
			00	not included	
			01; 10; 11	included	
21:20	BOOT_SEED_CUST_CFG	R/W	include customer factory area (including keys) in BOOT SEED computation		0
			00	not included	
			01; 10; 11	included	
19:18	BOOT_SEED_INC_NXP_CFG	R/W	include NXP area in BOOT SEED computation		0
			00	not included	
			01; 10; 11	included	
17:16	SKIP_BOOT_SEED	R/W	skip boot seed calculation		0
			00	enable calculation	
			01; 10; 11	disable calculation	

Table 278. Secure boot configuration - offset: 14h...continued

Bit	Name	Access	Description and value		Reset value
15:14	DICE_INC_SEC_EPOCH	R/W	include security EPOCH in DICE computation as per PSA specification On MCX W23 security epoch consists of following fields in CFPA page: • IMAGE_KEY_REVOKE • ROTKH_REVOKE • VENDOR_USAGE • DCFG_CC_SOCU_NS_PIN • DCFG_CC_SOCU_NS_DFLT		0
			00	not included	
			01; 10; 11	included	
15:12	reserved	-	reserved These bits must be set to 00.		-
13:12	reserved	-	reserved Read value is undefined. Only zero must be written.		-
11:10	BLOCK_SET_KEY	R/W	Block PUF key code generation		0
			00	enable key code generation	
			01; 10; 11	disable key code generation	
9:8	TZM_IMAGE_TYPE	R/W	TrustZone-M mode		0
			00	TZM mode in image header	
			01	disable TZ-M Boots to NonSecure.	
			10	TZ-M enable Boots to secure mode.	
			11	preset TZM checker from image header	
7:6	SKIP_DICE	R/W	skip DICE computation		0
			00	enable DICE	
			01; 10; 11	disable DICE	
7:6	reserved	-	reserved These bits must be set to 11.		-
5:4	DICE_CUST_CFG	R/W	include customer factory area (including keys) in DICE computation		0
			00	not included	
			01; 10; 11	included	
5:4	reserved	-	reserved These bits must be set to 00.		-

Table 278. Secure boot configuration - offset: 14h...continued

Bit	Name	Access	Description and value	Reset value
3:2	DICE_INC_NXP_CFG	R/W	include NXP area in DICE computation	0
			00 not included	
			01; 10; 11 included	
3:2	reserved	-	reserved These bits must be set to 00.	-
1:0	RSA4K	R/W	Use RSA4096 keys only	0
			00 RSA2048 keys	
			01; 10; 11 RSA4096 keys	

45.1.2.2.7 Prince configuration (PRINCE_CFG) register

Table 279. Prince configuration - offset: 1Ch

Bit	Name	Access	Description and value	Reset value
31:14	reserved	-	reserved Read value is undefined. Only zero must be written.	-
13	GLK	R/W	global lock enable When set, the Prince APB system access to the control register or to any region-configuration register is ignored. This bit behaves the same as the SLK bit.	0
12:10	reserved	-	reserved Read value is undefined. Only zero must be written.	-
9	SLK	R/W	system lock enable When set, the Prince APB system access to the control register or to any region-configuration register is ignored. This bit behaves the same as the GLK bit.	0
8:6	reserved	R/W	reserved Read value is undefined. Only zero must be written.	-
5	GDE	R/W	global decryption enable	0
4:2	reserved	-	reserved Read value is undefined. Only zero must be written.	-
1	GEE	R/W	global encryption enable	0
0	reserved	R/W	reserved Read value is undefined. Only zero must be written.	0

45.1.2.2.8 Bitmap control for memory context 0 (BMAPCTX0) register

Contains a 32-bit bitmap that controls if the corresponding 8-KByte block of flash (out of a maximum size of 256 KBytes) is encrypted/decrypted within memory context 0 using KEY0.

Table 280. Bitmap control for memory context 0 - offset: 20h

Bit	Name	Access	Description and value	Reset value
31:0	BENn	R/W	block enable for encryption/decryption To enable encryption/decryption for the corresponding 8-KByte block n, set bit n.	0
			0b disable encryption/decryption	
			1b enable encryption/decryption	

45.1.2.2.9 Bitmap control for memory context 1 (BMAPCTX1) register

Contains a 32-bit bitmap that controls if the corresponding 8-KByte block of flash (out of a maximum size of 256 KBytes) is encrypted/decrypted within memory context 1 using KEY1.

Table 281. Bitmap control for memory context 1 - offset: 24h

Bit	Name	Access	Description and value	Reset value
31:0	BENn	R/W	block enable for encryption/decryption To enable encryption/decryption for the corresponding 8-KByte block n, set bit n.	0
			0b disable encryption/decryption	
			1b enable encryption/decryption	

45.1.2.2.10 Bitmap control for memory context 2 (BMAPCTX2) register

Contains a 32-bit bitmap that controls if the corresponding 8-KByte block of flash (out of a maximum size of 256 KBytes) is encrypted/decrypted within memory context 2 using KEY2.

Table 282. Bitmap control for memory context 2 - offset: 28h

Bit	Name	Access	Description and value	Reset value
31:0	BENn	R/W	block enable for encryption/decryption To enable encryption/decryption for the corresponding 8-KByte block n, set bit n.	0
			0b disable encryption/decryption	
			1b enable encryption/decryption	

45.1.2.2.11 Bitmap control for memory context 3 (BMAPCTX3) register

Contains a 32-bit bitmap that controls if the corresponding 8-KByte block of flash (out of a maximum size of 256 KBytes) is encrypted/decrypted within memory context 3 using KEY3.

Table 283. Bitmap control for memory context 3 - offset: 28h

Bit	Name	Access	Description and value	Reset value
31:0	BENn	R/W	block enable for encryption/decryption To enable encryption/decryption for the corresponding 8-KByte block n, set bit n.	0
			0b disable encryption/decryption	
			1b enable encryption/decryption	

45.1.2.2.12 Flash remap size (FLASH_REMAP_SIZE) register

This 32-bit register contains the size of the image to remap, in bytes. The 12 LSBs are ignored, so the size granularity is 4 KB.

Table 284. Flash remap size - offset: 58h

Bit	Name	Access	Description and value	Reset value
31:0	FLASH_REMAP_SIZE	R/W	flash remap 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 4 KB.	0

45.1.2.2.13 Flash remap offset (FLASH_REMAP_OFFSET) register

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 4 KB.

Table 285. Flash remap offset - offset: 5Ch

Bit	Name	Access	Description and value	Reset value
31:0	FLASH_REMAP_OFFSET	R/W	flash remap 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 4 KB.	0

45.1.2.2.14 Root of trust keys table hash (ROTKHa) register

Table 286. Flash remap offset - offset: 60h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	Root of trust keys table hash	0

45.1.2.2.15 Customer-defined CMPA fields (CUSTOMER_DEFINEDa)

For a = 0 to 87.

Table 287. Customer-defined CMPA fields - offset: 80h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	customer-defined CMPA fields	0

45.1.2.2.16 SHA256 digest for CMPA (SHA256_DIGESTa)

For a = 0 to 7.

Table 288. SHA256 digest for CMPA - offset: 1E0h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	256-bit SHA digest field This value is calculated over all data in the CMPA page from byte offset 000h to 1DFh	0

45.1.3 OEM KEY STORE

45.1.3.1 OEM KEY STORE memory map

Table 289. OEM KEY STORE base address: 0010_5000h

Name	Access	Offset	Description	Width (bits)	Reset value	Section
HEADER	R/W	0h	valid OEM key store header	32	0000_0000h	Section 45.1.3.2.1
SBKEY_HEADER0	R/W	10h	secure boot key code header 0	32	0000_0000h	Section 45.1.3.2.2
SBKEY_HEADER1	R/W	14h	secure boot key code header 1	32	0000_0000h	Section 45.1.3.2.3
SBKEY_KEY_CODE0 - SBKEY_KEY_CODE11	R/W	18h - 44h	secure boot key code body	32	0000_0000h	Section 45.1.3.2.4
USER_KEK_HEADER0	R/W	48h	user key encryption code header 0	32	0000_0000h	Section 45.1.3.2.5
USER_KEK_HEADER1	R/W	4Ch	user key encryption code header 1	32	0000_0000h	Section 45.1.3.2.6
USER_KEK_BODY0 - USER_KEK_BODY11	R/W	50h - 7Ch	user key encryption code body	32	0000_0000h	Section 45.1.3.2.7

45.1.3.2 Register descriptions

45.1.3.2.1 Valid OEM key store header (HEADER) register

Table 290. Valid OEM key store header - offset: 0h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	valid OEM key store header A value of 9595_9595h indicates that the OEM key store is programmed with valid keys.	0

45.1.3.2.2 Secure boot key code header 0 (SBKEY_HEADER0) register

Table 291. Secure boot key code header 0 - offset: 10h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	secure boot key code header	0

45.1.3.2.3 Secure boot key code header 1 (SBKEY_HEADER1) register

Table 292. Secure boot key code header 1 - offset: 14h

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved Read value is undefined. Only zero must be written.	-
29:24	SIZE	R/W	key size	0
23:12	reserved	-	reserved Read value is undefined. Only zero must be written.	-
11:8	INDEX	R/W	key index	0
7:2	reserved	-	reserved Read value is undefined. Only zero must be written.	-
1:0	TYPE	R/W	key type	0

45.1.3.2.4 Secure boot key code body (SBKEY_BODYa) register

For a = 0 to 11

Table 293. Secure boot key code body - offset: 18h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	secure boot key code body	0

45.1.3.2.5 User key encryption code header 0 (USER_KEK_HEADER0) register

Table 294. User key encryption code header 0 - offset: 48h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	-	user key encryption code header	0

45.1.3.2.6 User key encryption code header 1 (USER_KEK_HEADER1) register

Table 295. User key encryption code header 1 - offset: 4Ch

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved Read value is undefined. Only zero must be written.	-
29:24	SIZE	R/W	user KEK size	0
23:12	reserved	-	reserved Read value is undefined. Only zero must be written.	-
11:8	INDEX	R/W	user KEK index	0
7:2	reserved	-	reserved Read value is undefined. Only zero must be written.	-
1:0	TYPE	R/W	user KEK type	0

45.1.3.2.7 User key encryption code body (USER_KEK_BODYa) register

For a = 0 to 11

Table 296. User key encryption code body - offset: 50h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	user key encryption code body	0

45.1.4 NXP KEY STORE

45.1.4.1 NXP KEY STORE memory map

Table 297. OEM KEY STORE base address: 0010_7800h

Name	Access	Offset	Description	Width (bits)	Reset value	Section
HEADER	R/W	0h	valid key store header	32	0000_0000h	Section 45.1.4.2.1
PUF_DISCHARGE_TIME	R/W	4h	PUF discharge time	32	0000_0000h	Section 45.1.4.2.2
ACTIVATION_CODE0 - ACTIVATION_CODE297	R/W	8h - 4ACh	PUF activation code	32	0000_0000h	Section 45.1.4.2.3
PRINCE_REGION0_HEADER0	R/W	4B0h	Prince region 0 key code header 0	32	0000_0000h	Section 45.1.4.2.4
PRINCE_REGION0_HEADER1	R/W	4B4h	Prince region 0 key code header 1	32	0000_0000h	Section 45.1.4.2.5
PRINCE_REGION0_BODY0 - PRINCE_REGION0_BODY11	R/W	4B8h - 4E4h	Prince region 0 key code body	32	0000_0000h	Section 45.1.4.2.6
PRINCE_REGION1_HEADER0	R/W	4E8h	Prince region 1 key code header 0	32	0000_0000h	Section 45.1.4.2.7
PRINCE_REGION1_HEADER1	R/W	4ECh	Prince region 1 key code header 1	32	0000_0000h	Section 45.1.4.2.8
PRINCE_REGION1_BODY0 - PRINCE_REGION1_BODY11	R/W	4F0h - 51Ch	Prince region 1 key code body	32	0000_0000h	Section 45.1.4.2.9
PRINCE_REGION2_HEADER0	R/W	520h	Prince region 2 key code header 0	32	0000_0000h	Section 45.1.4.2.10
PRINCE_REGION2_HEADER1	R/W	524h	Prince region 2 key code header 1	32	0000_0000h	Section 45.1.4.2.11
PRINCE_REGION2_BODY0 - PRINCE_REGION2_BODY11	R/W	528h - 554h	Prince region 2 key code body	32	0000_0000h	Section 45.1.4.2.12
PRINCE_REGION3_HEADER0	R/W	558h	Prince region 3 key code header 0	32	0000_0000h	Section 45.1.4.2.13
PRINCE_REGION3_HEADER1	R/W	55Ch	Prince region 3 key code header 1	32	0000_0000h	Section 45.1.4.2.14
PRINCE_REGION3_BODY0 - PRINCE_REGION3_BODY11	R/W	560h - 58Ch	Prince region 3 key code body	32	0000_0000h	Section 45.1.4.2.15
UDS_HEADER0	R/W	590h	UDS key code header 0	32	0000_0000h	Section 45.1.4.2.16
UDS_HEADER1	R/W	594h	UDS key code header 1	32	0000_0000h	Section 45.1.4.2.17
UDS_BODY0 - UDS_BODY11	R/W	598h - 5C4h	UDS key code body	32	0000_0000h	Section 45.1.4.2.18
NXP_DEVICE_PRIVATE_KEY0	R/W	60Ch - 63Ch	NXP device private key	32	0000_0000h	Section 45.1.4.2.19

Table 297. OEM KEY STORE base address: 0010_7800h...continued

Name	Access	Offset	Description	Width (bits)	Reset value	Section
NXP_DEVICE_CERTIFICATE_00 - NXP_DEVICE_CERTIFICATE_03	R/W	640h - 64Ch	NXP device certificate part 0	32	0000_0000h	Section 45.1.4.2.20
NXP_DEVICE_CERTIFICATE_10 - NXP_DEVICE_CERTIFICATE_13	R/W	650h - 65Ch	NXP device certificate part 1	32	0000_0000h	Section 45.1.4.2.21
NXP_DEVICE_CERTIFICATE_20 - NXP_DEVICE_CERTIFICATE_23	R/W	660h - 66Ch	NXP device certificate part 2	32	0000_0000h	Section 45.1.4.2.22
NXP_DEVICE_CERTIFICATE_30 - NXP_DEVICE_CERTIFICATE_33	R/W	670h - 67Ch	NXP device certificate part 3	32	0000_0000h	Section 45.1.4.2.23
NXP_ATTEST_PRIVATE_KEY0 - NXP_ATTEST_PRIVATE_Key12	R/W	68Ch - 6BCh	NXP attest private key	32	0000_0000h	Section 45.1.4.2.24
NXP_ATTEST_CERTIFICATE_00 - NXP_ATTEST_CERTIFICATE_03	R/W	6C0h - 6DCh	NXP attestation certificate (ECDSA_sign - r[255:128])	32	0000_0000h	Section 45.1.4.2.25
NXP_ATTEST_CERTIFICATE_10 - NXP_ATTEST_CERTIFICATE_13	R/W	6D0h - 6DCh	NXP attestation certificate (ECDSA_sign - r[127:0])	32	0000_0000h	Section 45.1.4.2.26
NXP_ATTEST_CERTIFICATE_20 - NXP_ATTEST_CERTIFICATE_23	R/W	6E0h - 6ECh	NXP attestation certificate (ECDSA_sign - s[255:128])	32	0000_0000h	Section 45.1.4.2.27
NXP_ATTEST_CERTIFICATE_30 - NXP_ATTEST_CERTIFICATE_33	R/W	6F0h - 6FCh	NXP attestation certificate (ECDSA_sign - s[127:0])	32	0000_0000h	Section 45.1.4.2.28
SHA256_DIGEST0 - SHA256_DIGEST7	R/W	7E0h - 7FCh	SHA-256 figest for NXP key store area	32	0000_0000h	Section 45.1.4.2.29

45.1.4.2 Register descriptions

45.1.4.2.1 Valid NXP key store header (HEADER) register

Table 298. Valid NXP key store header - offset: 0h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	valid NXP key store header A value of 9595_9595h indicates that the NXP key store is programmed with valid keys.	0

45.1.4.2.2 PUF discharge time (PUF_DISCHARGE_TIME)

Table 299. PUF discharge time - offset: 4h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	PUF discharge time in milliseconds	0

45.1.4.2.3 PUF activation code (ACTIVATION_CODEa)

For a = 0 to 297

Table 300. PUF activation code - offset: 8h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	PUF activation code	0

45.1.4.2.4 Prince region 0 key header 0 (PRINCE_REGION0_HEADER0)

Table 301. Prince region 0 key header 0 - offset: 4B0h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	prince region 0 key code header	0

45.1.4.2.5 Prince region 0 key header 1 (PRINCE_REGION0_HEADER1)

Table 302. Prince region 0 key header 1 - offset: 4B4h

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved Read value is undefined, only zero must be written.	-
29:24	SIZE	R/W	prince region 0 key code size	0
23:12	reserved	-	reserved Read value is undefined, only zero must be written.	-
11:8	INDEX	R/W	prince region 0 key code index	0
7:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1:0	TYPE	R/W	prince region 0 key code type	0

45.1.4.2.6 Prince region 0 key code body (PRINCE_REGION0_BODYa)

For a = 0 to 11

Table 303. Prince region 0 key header 1 - offset: 4B8h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	prince region 0 key body	0

45.1.4.2.7 Prince region 1 key header 0 (PRINCE_REGION1_HEADER0)

Table 304. Prince region 1 key header 0 - offset: 4E8h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	prince region 1 key code header	0

45.1.4.2.8 Prince region 1 key header 1 (PRINCE_REGION1_HEADER1)

Table 305. Prince region 1 key header 1 - offset: 4ECh

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved Read value is undefined, only zero must be written.	-
29:24	SIZE	R/W	prince region 1 key code size	0
23:12	reserved	-	reserved Read value is undefined, only zero must be written.	-
11:8	INDEX	R/W	prince region 1 key code index	0
7:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1:0	TYPE	R/W	prince region 1 key code type	0

45.1.4.2.9 Prince region 1 key code body (PRINCE_REGION1_BODYa)

For a = 0 to 11

Table 306. Prince region 1 key code body - offset: 4F0h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	prince region 1 key body	0

45.1.4.2.10 Prince region 2 key header 0 (PRINCE_REGION2_HEADER0)

Table 307. Prince region 2 key header 0 - offset: 520h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	prince region 2 key code header	0

45.1.4.2.11 Prince region 2 key header 1 (PRINCE_REGION2_HEADER1)

Table 308. Prince region 2 key header 1 - offset: 524h

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved Read value is undefined, only zero must be written.	-
29:24	SIZE	R/W	prince region 2 key code size	0
23:12	reserved	-	reserved Read value is undefined, only zero must be written.	-
11:8	INDEX	R/W	prince region 2 key code index	0
7:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1:0	TYPE	R/W	prince region 2 key code type	0

45.1.4.2.12 Prince region 2 key code body (PRINCE_REGION2_BODYa)

For a = 0 to 11

Table 309. Prince region 2 key code body - offset: 528h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	prince region 2 key body	0

45.1.4.2.13 Prince region 3 key header 0 (PRINCE_REGION3_HEADER0)

Table 310. Prince region 3 key header 0 - offset: 558h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	prince region 3 key code header	0

45.1.4.2.14 Prince region 3 key header 1 (PRINCE_REGION3_HEADER1)

Table 311. Prince region 3 key header 1 - offset: 55Ch

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved Read value is undefined, only zero must be written.	-
29:24	SIZE	R/W	prince region 3 key code size	0
23:12	reserved	-	reserved Read value is undefined, only zero must be written.	-
11:8	INDEX	R/W	prince region 3 key code index	0
7:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1:0	TYPE	R/W	prince region 3 key code type	0

45.1.4.2.15 Prince region 3 key code body (PRINCE_REGION3_BODYa)

For a = 0 to 11

Table 312. Prince region 3 key code body - offset: 560h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	prince region 3 key body	0

45.1.4.2.16 UDS key header 0 (UDS_HEADER0)

For a = 0 to 11

Table 313. UDS key header 0 - offset: 590h

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	UDS key code header	0

45.1.4.2.17 UDS key header 1 (UDS_HEADER1)

Table 314. UDS key header 1 - offset: 594h

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved Read value is undefined, only zero must be written.	-
29:24	SIZE	R/W	UDS code size	0
23:12	reserved	-	reserved Read value is undefined, only zero must be written.	-
11:8	INDEX	R/W	UDS key code index	0
7:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1:0	TYPE	R/W	UDS key code type	0

45.1.4.2.18 UDS key body (UDS_BODYa)

For a = 0 to 11

Table 315. UDS key body - offset: 598h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	UDS key code body	0

45.1.4.2.19 NXP device private key (NXP_DEVICE_PRIVATE_KEYa)

For a = 0 to 12

Table 316. NXP device private key - offset: 60Ch + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	NXP device private key	0

**45.1.4.2.20 NXP device certificate part 0
(NXP_DEVICE_CERTIFICATE_00 - NXP_DEVICE_CERTIFICATE_03)****Table 317. Offset**

Register	Offset
NXP_CERTIFICATE_00	640h
NXP_CERTIFICATE_01	644h
NXP_CERTIFICATE_02	648h
NXP_CERTIFICATE_03	64Ch

Table 318. NXP device certificate part 0

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	NXP device certificate part 0 - (ECDSA_sign - r[255:128])	0

**45.1.4.2.21 NXP device certificate part 1
(NXP_DEVICE_CERTIFICATE_10 - NXP_DEVICE_CERTIFICATE_13)****Table 319. Offset**

Register	Offset
NXP_CERTIFICATE_10	650h
NXP_CERTIFICATE_11	654h
NXP_CERTIFICATE_12	658h
NXP_CERTIFICATE_13	65Ch

Table 320. NXP device certificate part 1

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	NXP device certificate part 1 - (ECDSA_sign - r[127:0])	0

45.1.4.2.22 NXP device certificate part 2 (NXP_DEVICE_CERTIFICATE_20 - NXP_DEVICE_CERTIFICATE_23)

Table 321. Offset

Register	Offset
NXP_CERTIFICATE_20	660h
NXP_CERTIFICATE_21	664h
NXP_CERTIFICATE_22	668h
NXP_CERTIFICATE_23	66Ch

Table 322. NXP device certificate part 2

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	NXP device certificate part 2 - (ECDSA_sign - s[255:128])	0

45.1.4.2.23 NXP device certificate part 3 (NXP_DEVICE_CERTIFICATE_30 - NXP_DEVICE_CERTIFICATE_33)

Table 323. Offset

Register	Offset
NXP_CERTIFICATE_30	670h
NXP_CERTIFICATE_31	674h
NXP_CERTIFICATE_32	678h
NXP_CERTIFICATE_33	67Ch

Table 324. NXP device certificate part 3

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	NXP Device Certificate part 3 - (ECDSA_sign - s[127:0])	0

45.1.4.2.24 NXP attestation private key (NXP_ATTEST_PRIVATE_KEYa)

For a = 0 to 12

Table 325. NXP attestation private key - offset: 68Ch + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	NXP attestation private key	0

45.1.4.2.25 NXP attestation certificate part 0 (NXP_ATTEST_CERTIFICATE_00 - NXP_ATTEST_CERTIFICATE_03)

Table 326. Offset

Register	Offset
NXP_ATTEST_CERTIFICATE_00	6C0h
NXP_ATTEST_CERTIFICATE_01	6C4h
NXP_ATTEST_CERTIFICATE_02	6C8h
NXP_ATTEST_CERTIFICATE_03	6CCh

Table 327. NXP attestation certificate part 0

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	NXP attestation certificate part 0 - (ECDSA_sign - r[255:128])	0

45.1.4.2.26 NXP attestation certificate part 1 (NXP_ATTEST_CERTIFICATE_10 - NXP_ATTEST_CERTIFICATE_13)

Table 328. Offset

Register	Offset
NXP_ATTEST_CERTIFICATE_10	6D0h
NXP_ATTEST_CERTIFICATE_11	6D4h
NXP_ATTEST_CERTIFICATE_12	6D8h
NXP_ATTEST_CERTIFICATE_13	6DCh

Table 329. NXP attestation certificate part 1

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	NXP attestation certificate part 1 - (ECDSA_sign - r[127:0])	0

45.1.4.2.27 NXP attestation certificate part 2 (NXP_ATTEST_CERTIFICATE_20 - NXP_ATTEST_CERTIFICATE_23)

Table 330. Offset

Register	Offset
NXP_ATTEST_CERTIFICATE_20	6E0h
NXP_ATTEST_CERTIFICATE_21	6E4h
NXP_ATTEST_CERTIFICATE_22	6E8h
NXP_ATTEST_CERTIFICATE_23	6ECh

Table 331. NXP attestation certificate part 2

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	NXP attestation certificate part 2 - (ECDSA_sign - s[255:128])	0

45.1.4.2.28 NXP attestation certificate part 3 (NXP_ATTEST_CERTIFICATE_30 - NXP_ATTEST_CERTIFICATE_33)

Table 332. Offset

Register	Offset
NXP_ATTEST_CERTIFICATE_30	6F0h
NXP_ATTEST_CERTIFICATE_31	6F4h
NXP_ATTEST_CERTIFICATE_32	6F8h
NXP_ATTEST_CERTIFICATE_33	6FCh

Table 333. NXP attestation certificate part 3

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	NXP attestation certificate part 3 - (ECDSA_sign - s[127:0])	0

45.1.4.2.29 SHA256 digest for CMPA (SHA256_DIGESTa)

For a = 0 to 7.

Table 334. SHA256 digest for CMPA - offset: 7E0h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	256-bit SHA digest field This value is calculated over all data in the NXP key store area page from byte offset 000h to 1DFh	0

45.1.5 ROM PATCH

45.1.5.1 ROMPATCH memory map

FLASH_ROMPATCH base address: 10_8000h

Table 335. FLASH_ROMPATCH base address: 10_8000h

Name	Access	Offset	Description	Width (bits)	Reset value	Section
HEADER	R/W	0h	ROM patch header	32	0000_0000h	Section 45.1.5.2.1
PATCH0 - PATCH638	R/W	4h - 9FCh	ROM patch code	32	0000_0000h	Section 45.1.5.2.2

45.1.5.2 Register descriptions

45.1.5.2.1 ROM patch header (HEADER)

Table 336. ROM patch header - offset: 0h

Bit	Name	Access	Description and value	Reset value
31:24	IDENTIFIER	R/W	ROM patch identifier	0
23:16	TYPE	R/W	ROM patch type	0
15:8	SUB_TYPE	R/W	ROM patch subtype	0
7:0	ENTRIES	R/W	number of ROM patch entries	0

45.1.5.2.2 ROM patch code (PATCHa)

For a = 0 to 638

Table 337. ROM patch code - offset: 4h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	PATCH	R/W	ROM patch code	0

45.1.6 NMPA

The NMPA contains all the trim data and configuration fields.

45.1.6.1 NMPA memory map

FLASH_NMPA base address: 10_8000h

Table 338. FLASH_NMPA base address: 10_8000h

Name	Access	Offset	Description	Width (bits)	Section
TRNG_CONTROL	R	B00h	true random generator control	32	Section 45.1.6.2.1
TRNG_PKRRNG	R	B04h	true random generator poker range	32	Section 45.1.6.2.2
TRNG_PKRMX	R	B08h	true random number generator poker maximum limit	32	Section 45.1.6.2.3
TRNG_SDCTL	R	B0Ch	true random number generator seed control	32	Section 45.1.6.2.4
TRNG_SBLIM	R	B10h	true random number generator sparse bit limit	32	Section 45.1.6.2.5
TRNG_FRQMIN	R	B14h	true random number generator frequency count minimum limit	32	Section 45.1.6.2.6
TRNG_FRQMAX	R	B18h	true random number generator frequency count maximum limit	32	Section 45.1.6.2.7
TRNG_SCML	R	B1Ch	true random number generator statistical check monobit limit	32	Section 45.1.6.2.8
TRNG_SCR1L	R	B20h	true random number generator statistical check run length 1	32	Section 45.1.6.2.9
TRNG_SCR2L	R	B24h	true random number generator statistical check run length 2	32	Section 45.1.6.2.10
TRNG_SCR3L	R	B28h	true random number generator statistical check run length 3	32	Section 45.1.6.2.11
TRNG_SCR4L	R	B2Ch	true random number generator statistical check run length 4	32	Section 45.1.6.2.12
TRNG_SCR5L	R	B30h	true random number generator statistical check run length 5	32	Section 45.1.6.2.13
TRNG_SCR6PL	R	B34h	true random number generator statistical check run length 6+	32	Section 45.1.6.2.14
DEVICE_TYPE	R	B60h	device type and name	32	Section 45.1.6.2.15

Table 338. FLASH_NMPA base address: 10_8000h...continued

Name	Access	Offset	Description	Width (bits)	Section
UUID_ARRAY0 - UUID_ARRAY3	R	B70h - B7Ch	unique device identifier	32	Section 45.1.6.2.16
PERIPHENCFG	R	B80h	peripheral enable configuration	32	Section 45.1.6.2.17
RAMSIZECFG	R	B84h	RAM size configuration	32	Section 45.1.6.2.18
FLASHSIZECFG	R	B88h	flash size configuration	32	Section 45.1.6.2.19
DIS_ROM_HIDING	R	B8Ch	disable ROM hiding	32	Section 45.1.6.2.20
RINGO_0	R	B90h	ring oscillator 0 control configuration	32	Section 45.1.6.2.21
RINGO_1	R	B94h	ring oscillator 1 control configuration	32	Section 45.1.6.2.22
RINGO_2	R	B98h	ring oscillator 2 control configuration	32	Section 45.1.6.2.23
PUF_SRAM	R	B9Ch	PUF SRAM control configuration	32	Section 45.1.6.2.24
BOD	R	BA0h	BOD configuration	32	Section 45.1.6.2.25
LDO_MEM	R	BA4h	memory LDO configuration	32	Section 45.1.6.2.26
LDO_DIG	R/W	BA8h	digital LDO configuration	32	Section 45.1.6.2.27
DCDC_POWER_PROFILE_HIGH	R	BC8h	DC-to-DC high power profile	32	Section 45.1.6.2.28
SHA256_DIGEST0 - SHA256_DIGEST7	R	BE0h - BFCh	SHA-256 digest for ROM PATCH and NMPA area	32	Section 45.1.6.2.29
FMUPARM0_0	R	1600h	general FMU parameters 0	32	Section 45.1.6.2.30
FMUPARM0_1	R	1604h	general FMU parameters 1	32	Section 45.1.6.2.31
FMUPARM0_2	R	1608h	general FMU parameters 2	32	Section 45.1.6.2.32
FMUPARM0_3	R	160Ch	general FMU parameters 3	32	Section 45.1.6.2.33
FMUPARM1_0	R	1610h	SMW timer options, SMW setting options 0	32	Section 45.1.6.2.34
FMUPARM1_1	R	1614h	SMW timer options, SMW setting options 1	32	Section 45.1.6.2.35
FMUPARM1_2	R	1618h	SMW timer options, SMW setting options 2	32	Section 45.1.6.2.36
FMUPARM1_3	R	161Ch	SMW timer options, SMW setting options 3	32	Section 45.1.6.2.37

Table 338. FLASH_NMPA base address: 10_8000h...continued

Name	Access	Offset	Description	Width (bits)	Section
FMUPARM2_0	R	1620h	SMW setting options continued, SMW smart program/erase options 0	32	Section 45.1.6.2.38
FMUPARM2_1	R	1624h	SMW setting options continued, SMW smart program/erase options 1	32	Section 45.1.6.2.39
FMUPARM2_2	R	1628h	SMW setting options continued, SMW smart program/erase options 2	32	Section 45.1.6.2.40
FMUPARM2_3	R	162Ch	SMW setting options continued, SMW smart program/erase options 3	32	Section 45.1.6.2.41
FMUPARM3	R	1630h	SMW smart erase options continued	32	Section 45.1.6.2.42
FMUPARM4	R	1640h	array size (see MSIZE register)	32	Section 45.1.6.2.43
FMUPARM5	R	1650h	TME control (see FTEST register)	32	Section 45.1.6.2.44
SOCTRIMEN	R	17E0h	SoC trim enable	32	Section 45.1.6.2.45
SOCTRIMECC	R	17F0h	SoC trim ECC decoder enable	32	Section 45.1.6.2.46
GPO0_0	R	1800h	SOCTRIM0 FRO32K	32	Section 45.1.6.2.47
GPO0_2	R	1808h	SOCTRIM0 flash speed	32	Section 45.1.6.2.48
GPO1_0	R	1810h	SOCTRIM1 device ID	32	Section 45.1.6.2.49
GPO1_1	R	1814h	SOCTRIM1 device revision	32	Section 45.1.6.2.50
GPO2_0	R	1820h	SOCTRIM2 FRO192M	32	Section 45.1.6.2.51
GPO3_0	R	1830h	SOCTRIM3 band gap	32	Section 45.1.6.2.52

45.1.6.2 Register descriptions

45.1.6.2.1 True random number generator control (TRNG_CONTROL)

Table 339. True random number generator control - offset: B00h

Bit	Name	Access	Description and value
31	TRNG_VALID	R	Indicates if the other TRNG values in flash are valid and can be copied to the HW block by the ROM bootloader.
30	reserved	-	reserved Read value is undefined, only zero must be written.
29	TRNG_SEC_CFG_NO_PROG	R	Value to be copied to the NO_PRGM field in the SEC_CFG register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.
28:20	reserved	-	reserved Read value is undefined, only zero must be written.
19:16	TRNG_SCMISC_RTY_CT	R	Value to be copied to the RTY_CT field in the SCMISC register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.
15:8	TRNG_SCMISC_LRUN_MAX	R	Value to be copied to the LRUN field in the SCMISC register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.
7	TRNG_MCTL_FOR_SCLK	R	Value to be copied to the FOR_SCLK field in the MCTL register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.
6:4	reserved	-	reserved Read value is undefined, only zero must be written.
3:2	TRNG_MCTL_OSC_DIV	R	Value to be copied to the OSC_DIV field in the MCTL register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.
1:0	TRNG_MCTL_SAMP_MODE	R	Value to be copied to the SAMP_MODE field in the MCTL register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.

45.1.6.2.2 True random number generator poker range (TRNG_PKRRNG)

Table 340. True random number generator poker range - offset: B04h

Bit	Name	Access	Description and value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.
15:0	TRNG_PKR_RNG	R	Value to be copied to the PKR_RNG field in the PKRRNG register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.

45.1.6.2.3 True random number generator poker maximum limit (TRNG_PKRMX)

Table 341. True random number generator poker maximum limit - offset: B08h

Bit	Name	Access	Description and value
31:24	reserved	-	reserved Read value is undefined, only zero must be written.
23:0	TRNG_PKR_MAX	R	Value to be copied to the PKR_MAX field in the PKRMX register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.

45.1.6.2.4 True random number generator seed control (TRNG_SDCTL)

Table 342. True random number generator seed control - offset: B0Ch

Bit	Name	Access	Description and value
31:16	TRNG_ENT_DLY	R	Value to be copied to the ENT_DLY field in the SDCTL register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.
15:0	TRNG_SAMP_SIZE	R	Value to be copied to the SAMP_SIZE field in the SDCTL register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.

45.1.6.2.5 True random number generator sparse bit limit (TRNG_SBLIM)

Table 343. True random number generator sparse bit limit - offset: B10h

Bit	Name	Access	Description and value
31:10	reserved	--	reserved Read value is undefined, only zero must be written.
9:0	TRNG_SB_LIM	R	Value to be copied to the SB_LIM field in the SBLIM register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.

45.1.6.2.6 True random number generator frequency count minimum limit (TRNG_FRQMIN)

Table 344. True random number generator frequency count minimum limit - offset: B14h

Bit	Name	Access	Description and value
31:22	reserved	--	reserved Read value is undefined, only zero must be written.
21:0	TRNG_FRQ_MIN	R	Value to be copied to the FRQ_MIN field in the FRQMIN register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.

45.1.6.2.7 True random number generator frequency count maximum limit (TRNG_FRQMAX)

Table 345. True random number generator frequency count maximum limit - offset: B18h

Bit	Name	Access	Description and value
31:22	reserved	--	reserved Read value is undefined, only zero must be written.
21:0	TRNG_FRQ_MAX	R	Value to be copied to the FRQ_MAX field in the FRQMAX register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.

45.1.6.2.8 True random number generator poker range statistical check monobit limit (TRNG_SCML)

Table 346. True random number generator poker range statistical check monobit limit - offset: B1Ch

Bit	Name	Access	Description and value
31:16	TRNG_MONO_RNG	R	Value to be copied to the MONO_RNG field in the SCML register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.
15:0	TRNG_MONO_MAX	R	Value to be copied to the MONO_MAX field in the SCML register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.

45.1.6.2.9 True random number generator poker range statistical check run length 1 limit (TRNG_SCR1L)

Table 347. True random number generator poker range statistical check run length 1 limit - offset: B20h

Bit	Name	Access	Description and value
31	reserved	-	reserved Read value is undefined, only zero must be written.
30:16	TRNG_RUN1_RNG	R	Value to be copied to the RUN1_RNG field in the SCR1L register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.
15	reserved	-	reserved Read value is undefined, only zero must be written.
14:0	TRNG_RUN1_MAX	R	Value to be copied to the RUN1_MAX field in the SCR1L register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.

45.1.6.2.10 True random number generator poker range statistical check run length 2 limit (TRNG_SCR2L)

Table 348. True random number generator poker range statistical check run length 2 limit - offset: B24h

Bit	Name	Access	Description and value
31:30	reserved	-	reserved Read value is undefined, only zero must be written.
29:16	TRNG_RUN2_RNG	R	Value to be copied to the RUN2_RNG field in the SCR2L register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.
15:14	reserved	-	reserved

Table 348. True random number generator poker range statistical check run length 2 limit - offset: B24h...continued

Bit	Name	Access	Description and value
			Read value is undefined, only zero must be written.
13:0	TRNG_RUN2_MAX	R	Value to be copied to the RUN2_MAX field in the SCR2L register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.

45.1.6.2.11 True random number generator poker range statistical check run length 3 limit (TRNG_SCR3L)

Table 349. True random number generator poker range statistical check run length 3 limit - offset: B28h

Bit	Name	Access	Description and value
31:29	reserved	-	reserved Read value is undefined, only zero must be written.
28:16	TRNG_RUN3_RNG	R	Value to be copied to the RUN3_RNG field in the SCR3L register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.
15:13	reserved	-	reserved Read value is undefined, only zero must be written.
12:0	TRNG_RUN3_MAX	R	Value to be copied to the RUN3_MAX field in the SCR3L register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.

45.1.6.2.12 True random number generator poker range statistical check run length 4 limit (TRNG_SCR4L)

Table 350. True random number generator poker range statistical check run length 4 limit - offset: B2Ch

Bit	Name	Access	Description and value
31:28	reserved	-	reserved Read value is undefined, only zero must be written.
27:16	TRNG_RUN4_RNG	R	Value to be copied to the RUN4_RNG field in the SCR4L register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.
15:12	reserved	-	reserved Read value is undefined, only zero must be written.
11:0	TRNG_RUN4_MAX	R	Value to be copied to the RUN4_MAX field in the SCR4L register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.

45.1.6.2.13 True random number generator poker range statistical check run length 5 limit (TRNG_SCR5L)

Table 351. True random number generator poker range statistical check run length 5 limit - offset: B30h

Bit	Name	Access	Description and value
31:27	reserved	-	reserved Read value is undefined, only zero must be written.

Table 351. True random number generator poker range statistical check run length 5 limit - offset: B30h...continued

Bit	Name	Access	Description and value
26:16	TRNG_RUN5_RNG	R	Value to be copied to the RUN5_RNG field in the SCR5L register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.
15:11	reserved	-	reserved Read value is undefined, only zero must be written.
10:0	TRNG_RUN5_MAX	R	Value to be copied to the RUN5_MAX field in the SCR5L register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.

45.1.6.2.14 True random number generator poker range statistical check run length 6+ limit (TRNG_SCR6PL)

Table 352. True random number generator poker range statistical check run length 6+ limit - offset: B34h

Bit	Name	Access	Description and value
31:27	reserved	-	reserved Read value is undefined, only zero must be written.
26:16	TRNG_RUN6P_RNG	R	Value to be copied to the RUN6P_RNG field in the SCR6PL register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.
15:11	reserved	-	reserved Read value is undefined, only zero must be written.
10:0	TRNG_RUN6P_MAX	R	Value to be copied to the RUN6P_MAX field in the SCR6PL register of the TRNG module by the ROM bootloader when TRNG_VALID bit equals 1b.

45.1.6.2.15 Device name and type (DEVICE_TYPE)

Table 353. Device name and type - offset: B60h

Bit	Name	Access	Description and value
31:24	DEVICE_TYPE_PIN	R	number of pins on the package (in hexadecimal format).
23:20	DEVICE_TYPE_PKG	R	device package type
			0010 HVQFN
			1000 WLSCP
19:17	reserved	-	reserved Read value is undefined, only zero must be written.
16	DEVICE_TYPE_SEC	R	security device type
			1 secure family
15:0	DEVICE_TYPE_NUM	R	device type number 0x236B for MCXW236B 0x235B for MCXW235B 0x236A for MCXW236A

45.1.6.2.16 Unique device identifier (UUID_ARRAY0 - UUID_ARRAY3)

Table 354. Offset

Register	Offset
UUID_ARRAY0	B70h
UUID_ARRAY1	B74h
UUID_ARRAY2	B78h
UUID_ARRAY3	B7Ch

Table 355. Unique device identifier - offset: B70h (a × 4h)

Bit	Name	Access	Description and value
31:0	FIELD	R	128-bit unique device identifier

45.1.6.2.17 Peripheral enable configuration (PERIPHENCFG)

Table 356. Peripheral enable configuration - Offset: B80h

Bit	Name	Access	Description and value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.
15:0	PERIPHERAL_CONFIGURATION	R	Value to be copied to the PERIPHENCFG register of the SYSCON module by the ROM bootloader. Indicates which modules are enabled during the operation of the chip.

45.1.6.2.18 RAM size configuration (RAMSIZECFG)

Table 357. RAM size configuration - Offset: B84h

Bit	Name	Access	Description and value
31:0	SRAM_CONFIGURATION	R	Value to be copied to the RAMSIZECFG register of the SYSCON module by the ROM bootloader. Indicates the supported SRAM size of the chip.

45.1.6.2.19 Flash size configuration (FLASHSIZECFG)

Table 358. Flash size configuration - Offset: B88h

Bit	Name	Access	Description and value
31:0	FLASH_CONFIGURATION	R	Value to be copied to the FLASHSIZECFG register of the SYSCON module by the ROM bootloader. Indicates the supported flash size of the chip.

45.1.6.2.20 Disable ROM hiding (DIS_ROM_HIDING)

Table 359. Disable ROM hiding - Offset: B8Ch

Bit	Name	Access	Description and value
31:0	DIS_ROM_HIDING	R	When this register contains the value 3CC3_5AA5h, the ROM hiding feature is disabled. For all other values, the critical ROM is hidden.

45.1.6.2.21 Ring oscillator 0 control (RINGO_0)

Table 360. Ring oscillator 0 control - Offset: B90h

Bit	Name	Access	Description and value
31:1	RINGO_0_CTRL	R	Value to be copied to the RINGO0_CTRL register of the ANACTRL module by the ROM bootloader when RINGO_0_CTRL_VALID bit is set to 1b.
0	RINGO_0_CTRL_VALID	R	ring oscillator 0 control validation
			0 RINGO_0_CTRL field is not valid
			1 RINGO_0_CTRL field is valid

45.1.6.2.22 Ring oscillator 1 control (RINGO_1)

Table 361. Ring oscillator 1 control - Offset: B94h

Bit	Name	Access	Description and value
31:1	RINGO_1_CTRL	R	Value to be copied to the RINGO1_CTRL register of the ANACTRL module by the ROM bootloader when RINGO_1_CTRL_VALID bit is set to 1b.
0	RINGO_1_CTRL_VALID	R	ring oscillator 1 control validation
			0 RINGO_1_CTRL field is not valid
			1 RINGO_1_CTRL field is valid

45.1.6.2.23 Ring oscillator 2 control (RINGO_2)

Table 362. Ring oscillator 2 control - Offset: B98h

Bit	Name	Access	Description and value
31:1	RINGO_2_CTRL	R	Value to be copied to the RINGO2_CTRL register of the ANACTRL module by the ROM bootloader when RINGO_2_CTRL_VALID bit is set to 1b.
0	RINGO_2_CTRL_VALID	R	ring oscillator 2 control validation
			0 RINGO_2_CTRL field is not valid
			1 RINGO_2_CTRL field is valid

45.1.6.2.24 PUF SRAM control configuration (PUF_SRAM)

Table 363. PUF SRAM control configuration - Offset: B9Ch

Bit	Name	Access	Description and value
31:24	reserved	-	reserved

Table 363. PUF SRAM control configuration - Offset: B9Ch...continued

Bit	Name	Access	Description and value
			Read value is undefined, only zero must be written.
23:21	WAM	R	Value to be copied to the WAM register field of the PUF module by the ROM bootloader when PUF_SRAM_VALID bit is set to 1b.
20:19	RAM	R	Value to be copied to the RAM register field of the PUF module by the ROM bootloader when PUF_SRAM_VALID bit is set to 1b.
18:16	WM	R	Value to be copied to the WM register field of the PUF module by the ROM bootloader when PUF_SRAM_VALID bit is set to 1b.
15:11	RM	R	Value to be copied to the RM register field of the PUF module by the ROM bootloader when PUF_SRAM_VALID bit is set to 1b.
10:8	SMB	R	Value to be copied to the SMB register field of the PUF module by the ROM bootloader when PUF_SRAM_VALID bit is set to 1b.
7:3	reserved	-	reserved Read value is undefined, only zero must be written.
2	CKGATING	R	Value to be copied to the CKGATING register field of the PUF module by the ROM bootloader when PUF_SRAM_VALID bit is set to 1b.
1	MODE	R	Value to be copied to the MODE register field of the PUF module by the ROM bootloader when PUF_SRAM_VALID bit is set to 1b.
0	PUF_SRAM_VALID	R	PUF SRAM fields validation
			0 PUF_SRAM fields in this register are not valid
			1 PUF_SRAM fields in this register are valid

45.1.6.2.25 Brownout detector (BOD)

Table 364. Brownout detector - Offset: BA0h

Bit	Name	Access	Description and value
31:26	reserved	-	reserved Read value is undefined, only zero must be written.
25	BOD2_LVLSEL	R	Value to be copied to the LVLSEL register field of the BOD2 module by the ROM bootloader when BOD2_TRIM_VALID bit is set to 1b.
24	BOD2_MODE	R	Value to be copied to the MODE register field of the BOD2 module by the ROM bootloader when BOD2_TRIM_VALID bit is set to 1b.
23:22	BOD2_HYST	R	Value to be copied to the HYST register field of the BOD2 module by the ROM bootloader when BOD2_TRIM_VALID bit is set to 1b.

Table 364. Brownout detector - Offset: BA0h...continued

Bit	Name	Access	Description and value
21:17	BOD2_TRIGLVL	R	Value to be copied to the TRIGLVL register field of the BOD2 module by the ROM bootloader when BOD2_TRIM_VALID bit is set to 1b.
16	BOD2_TRIM_VALID	R	brownout detector 2 fields validation
			0 BOD2 fields in this register are not valid
			1 BOD2 fields in this register are valid
15:10	reserved	-	reserved Read value is undefined, only zero must be written.
9	BOD1_LVLSEL	R	Value to be copied to the LVLSEL register field of the BOD1 module by the ROM bootloader when BOD1_TRIM_VALID bit is set to 1b.
8	BOD1_MODE	R	Value to be copied to the MODE register field of the BOD1 module by the ROM bootloader when BOD1_TRIM_VALID bit is set to 1b.
7:6	BOD1_HYST	R	Value to be copied to the HYST register field of the BOD1 module by the ROM bootloader when BOD1_TRIM_VALID bit is set to 1b.
5:1	BOD1_TRIGLVL	R	Value to be copied to the TRIGLVL register field of the BOD1 module by the ROM bootloader when BOD1_TRIM_VALID bit is set to 1b.
0	BOD1_TRIM_VALID	R	brownout detector fields validation
			0 BOD1 fields in this register are not valid
			1 BOD1 fields in this register are valid

45.1.6.2.26 LDO_MEM (LDO_MEM)

Table 365. LDO_MEM - Offset: BA4h

Bit	Name	Access	Description and value
31:30	reserved	-	reserved Read value is undefined, only zero must be written.
29:25	DPDW_TRIM	R	When DPDW_TRIM_VALID bit is set to 1b, this value is copied to the LDOMEM register field in the PMC before transitioning to the deep-power down state.
24	DPDW_TRIM_VALID	R	DPDW_TRIM field validation
			0 DPDW_TRIM field in this register is not valid
			1 DPDW_TRIM field in this register is valid
23:22	reserved	-	reserved Read value is undefined, only zero must be written.
21:17	PDWN_TRIM	R	When DPDW_TRIM_VALID bit is set to 1b, this value is copied to the LDOMEM register field in the PMC before transitioning to the power-down state.
16	PDWN_TRIM_VALID	R	PDWN_TRIM field validation
			0 DPDW_TRIM field in this register is not valid

Table 365. LDO_MEM - Offset: BA4h...continued

Bit	Name	Access	Description and value	
			1	DPDW_TRIM field in this register is valid
15:14	reserved	-	reserved Read value is undefined, only zero must be written.	
13:9	DSLP_TRIM	R	When DSLP_TRIM_VALID bit is set to 1b, this value is copied to the LDOMEM register field in the PMC before transitioning to the deep-sleep state.	
8	DSLP_TRIM_VALID	R	DSLP_TRIM field validation	
			0	DSLP_TRIM field in this register is not valid
			1	DSLP_TRIM field in this register is valid
7:6	reserved	-	reserved Read value is undefined, only zero must be written.	
5:1	ACTIVE_TRIM	R	When ACTIVE_TRIM_VALID bit is set to 1b, this value is copied to the LDOMEM register field in the PMC when transitioning out of a low-power state into the active state.	
0	ACTIVE_TRIM_VALID	R	ACTIVE_TRIM field validation	
			0	ACTIVE_TRIM field in this register is not valid
			1	ACTIVE_TRIM field in this register is valid

45.1.6.2.27 LDO_DIG (LDO_DIG)

Table 366. LDO_DIG - Offset: BA8h

Bit	Name	Access	Description and value	
31:30	reserved	-	reserved Read value is undefined, only zero must be written.	
29:25	DPDW_TRIM	R	When DPDW_TRIM_VALID bit is set to 1b, this value is copied to the LDODIG register field in the PMC before transitioning to the deep-power down state.	
24	DPDW_TRIM_VALID	R	DPDW_TRIM field validation	
			0	DPDW_TRIM field in this register is not valid
			1	DPDW_TRIM field in this register is valid
23:22	reserved	-	reserved Read value is undefined, only zero must be written.	
21:17	PDWN_TRIM	R	When DPDW_TRIM_VALID bit is set to 1b, this value is copied to the LDODIG register field in the PMC before transitioning to the power-down state.	
16	PDWN_TRIM_VALID	R	PDWN_TRIM field validation	
			0	DPDW_TRIM field in this register is not valid
			1	DPDW_TRIM field in this register is valid
15:14	reserved	-	reserved Read value is undefined, only zero must be written.	

Table 366. LDO_DIG - Offset: BA8h...continued

Bit	Name	Access	Description and value
13:9	DSLP_TRIM	R	When DSLP_TRIM_VALID bit is set to 1b, this value is copied to the LDODIG register field in the PMC before transitioning to the deep-sleep state.
8	DSLP_TRIM_VALID	R	DSLP_TRIM field validation
			0 DSLP_TRIM field in this register is not valid
			1 DSLP_TRIM field in this register is valid
7:6	reserved	-	reserved Read value is undefined, only zero must be written.
5:1	ACTIVE_TRIM	R	When ACTIVE_TRIM_VALID bit is set to 1b, this value is copied to the LDODIG register field in the PMC when transitioning out of a low-power state into the active state.
0	ACTIVE_TRIM_VALID	R	ACTIVE_TRIM field validation
			0 ACTIVE_TRIM field in this register is not valid
			1 ACTIVE_TRIM field in this register is valid

45.1.6.2.28 DCDC high power profile (DCDC_POWER_PROFILE_HIGH)

Table 367. DCDC high power profile - Offset: BC8h

Bit	Name	Access	Description and value
31:28	reserved	-	reserved Read value is undefined, only zero must be written.
27:24	VOUT_PWD	R	When the high power profile for the DCDC is selected and the DCDC_TRIM_VALID bit is set to 1b, this field is copied to the VOUT_PWD field of the DCDC register in the PMC module.
23	INDUCTORCLAMPENABLE	R	When the high power profile for the DCDC is selected and the DCDC_TRIM_VALID bit is set to 1b, this field is copied to the INDUCTORCLAMPENABLE field of the DCDC register in the PMC module.
22	SLICING_ENABLE	R	When the high power profile for the DCDC is selected and the DCDC_TRIM_VALID bit is set to 1b, this field is copied to the SLICINGENABLE field of the DCDC register in the PMC module.
21:18	VOUT	R	When the high power profile for the DCDC is selected and the DCDC_TRIM_VALID bit is set to 1b, this field is copied to the VOUT field of the DCDC register in the PMC module.
17	DISABLE_ISENSE	R	When the high power profile for the DCDC is selected and the DCDC_TRIM_VALID bit is set to 1b, this field is copied to the DISABLEISENSE field of the DCDC register in the PMC module.
16:12	TMOS	R	When the high power profile for the DCDC is selected and the DCDC_TRIM_VALID bit is set to 1b, this field is copied to the TMOS field of the DCDC register in the PMC module.
11	ICENABLE	R	When the high power profile for the DCDC is selected and the DCDC_TRIM_VALID bit is set to 1b, this field is copied to the ICENABLE field of the DCDC register in the PMC module.
10:9	ISEL	R	When the high power profile for the DCDC is selected and the DCDC_TRIM_VALID bit is set to 1b, this field is copied to the ISEL field of the DCDC register in the PMC module.
8:7	ICOMP	R	When the high power profile for the DCDC is selected and the DCDC_TRIM_VALID bit is set to 1b, this field is copied to the ICOMP field of the DCDC register in the PMC module.
6:1	RC	R	When the high power profile for the DCDC is selected and the DCDC_TRIM_VALID bit is set to 1b, this field is copied to the RC field of the DCDC register in the PMC module.
0	DCDC_TRIM_VALID	R	DCDC fields validation
			0 DCDC fields in this register are not valid.
			1 DCDC fields in this register are valid.

45.1.6.2.29 SHA-256 DIGEST for ROMPATCH and NMPA area (0000 - 0BEC and 1800 - 187C)(SHA256_DIGESTa)

For a = 0 to 7.

Table 368. SHA256 digest for CMPA - offset: BE0h + (a × 4h)

Bit	Name	Access	Description and value
31:0	FIELD	R	256-bit SHA digest field This value is calculated over all data in the ROMPATCH and NMPA area pages from byte offset 0000h to 0BEFh and from byte offset 1800h to 187Ch

45.1.6.2.30 General FMU parameters (FMUPARM0_0)

Table 369. General FMU parameters - offset: 1600h)

Bit	Name	Access	Description and value
31:0	FIELD	R	When this field is set to the value CODE_CODEh, the other FMU parameters will be loaded from IFR1 into the registers.

45.1.6.2.31 General FMU parameters (FMUPARM0_1)

Table 370. General FMU parameters - offset: 1604h)

Bit	Name	Access	Description and value
31:0	FIELD	R	When FMUPARM0_0 register is set to the value CODE_CODEh, the value in this field is copied to the following register fields in the flash controller: FCTRL[RWSC] = FMUPARM0_1[3:0]

45.1.6.2.32 General FMU parameters (FMUPARM0_2)

Table 371. General FMU parameters - offset: 1608h)

Bit	Name	Access	Description and value
31:0	FIELD	R	When FMUPARM0_0 register is set to the value CODE_CODEh, the value in this field is copied to the following register fields in the flash controller: - MCTL[LSACT_EN] = FMUPARM0_2[2] - MCTL[LSACTWREN] = FMUPARM0_2[3] - MCTL[MASTER_REPAIR_EN] = FMUPARM0_2[4] - MCTL[RFCMDEN] = FMUPARM0_2[5] - MCTL[CWSABTEN] = FMUPARM0_2[6] - MCTL[MRGRDDIS] = FMUPARM0_2[7] - MCTL[MRGRD0] = FMUPARM0_2[11:8] - MCTL[MRGRD1] = FMUPARM0_2[15:12] - MCTL[SALV_DIS] = FMUPARM0_2[22]

45.1.6.2.33 General FMU parameters (FMUPARM0_3)

Table 372. General FMU parameters - offset: 160Ch)

Bit	Name	Access	Description and value
31:0	FIELD	R	When FMUPARM0_0 register is set to the value CODE_CODEh, the value in this field is copied to the following register fields in the flash controller: - PWR_OPT[PD_CDIV] = FMUPARM0_3[7:0] - PWR_OPT[SLM_COUNT] = FMUPARM0_3[25:16]

Table 372. General FMU parameters - offset: 160Ch)

Bit	Name	Access	Description and value
			PWR_OPT[PD_TIMER_EN] = FMUPARM0_1[31]

45.1.6.2.34 SMW timer options, SMW setting options (FMUPARM1_0)

Table 373. SMW timer options, SMW setting options - offset: 1610h)

Bit	Name	Access	Description and value
31:0	FIELD	R	When FMUPARM0_0 register is set to the value C0DE_C0DEh, the value in this field is copied to the following register fields in the flash controller: SMW_TIMER_OPTION = FMUPARM1_0[12:0]

45.1.6.2.35 SMW timer options, SMW setting options (FMUPARM1_1)

Table 374. SMW timer options, SMW setting options - offset: 1614h)

Bit	Name	Access	Description and value
31:0	FIELD	R	When FMUPARM0_0 register is set to the value C0DE_C0DEh, the value in this field is copied to the following register fields in the flash controller: SMW_SETTING_OPTION0 = FMUPARM1_1[31:0]

45.1.6.2.36 SMW timer options, SMW setting options (FMUPARM1_2)

Table 375. SMW timer options, SMW setting options - offset: 1618h)

Bit	Name	Access	Description and value
31:0	FIELD	R	When FMUPARM0_0 register is set to the value C0DE_C0DEh, the value in this field is copied to the following register fields in the flash controller: SMW_SETTING_OPTION2 = FMUPARM1_2[31:0]

45.1.6.2.37 SMW timer options, SMW setting options (FMUPARM1_3)

Table 376. SMW timer options, SMW setting options - offset: 161Ch)

Bit	Name	Access	Description and value
31:0	FIELD	R	When FMUPARM0_0 register is set to the value C0DE_C0DEh, the value in this field is copied to the following register fields in the flash controller: SMW_SETTING_OPTION3 = FMUPARM1_3[18:0]

45.1.6.2.38 SMW setting options continued, SMW smart program/erase options (FMUPARM2_0)

Table 377. SMW setting options continued, SMW smart program/erase options - offset: 1620h)

Bit	Name	Access	Description and value
31:0	FIELD	R	When FMUPARM0_0 register is set to the value C0DE_C0DEh, the value in this field is copied to the following register fields in the flash controller: SMW_SMP_WHV_OPTION0 = FMUPARM2_0[31:0]

45.1.6.2.39 SMW setting options continued, SMW smart program/erase options (FMUPARM2_1)

Table 378. SMW setting options continued, SMW smart program/erase options - offset: 1624h)

Bit	Name	Access	Description and value
31:0	FIELD	R	When FMUPARM0_0 register is set to the value C0DE_C0DEh, the value in this field is copied to the following register fields in the flash controller: • SMW_SME_WHV_OPTION0 = FMUPARM2_1[31:0]

45.1.6.2.40 SMW setting options continued, SMW smart program/erase options (FMUPARM2_2)

Table 379. SMW setting options continued, SMW smart program/erase options - offset: 1628h)

Bit	Name	Access	Description and value
31:0	FIELD	R	When FMUPARM0_0 register is set to the value C0DE_C0DEh, the value in this field is copied to the following register fields in the flash controller: • SMW_SETTING_OPTION1 = FMUPARM2_2[31:0]

45.1.6.2.41 SMW setting options continued, SMW smart program/erase options (FMUPARM2_3)

Table 380. SMW setting options continued, SMW smart program/erase options - offset: 162Ch)

Bit	Name	Access	Description and value
31:0	FIELD	R	When FMUPARM0_0 register is set to the value C0DE_C0DEh, the value in this field is copied to the following register fields in the flash controller: • SMW_SMP_WHV_OPTION1 = FMUPARM2_2[31:0]

45.1.6.2.42 SMW smart erase options continued (FMUPARM3)

Table 381. SMW smart erase options continued - offset: 1630h)

Bit	Name	Access	Description and value
31:0	FIELD	R	When FMUPARM0_0 register is set to the value C0DE_C0DEh, the value in this field is copied to the following register fields in the flash controller: • SMW_SME_WHV_OPTION1 = FMUPARM3[31:0]

45.1.6.2.43 Flash array size (FMUPARM4)

Table 382. Flash array size - offset: 1640h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R	When FMUPARM0_0 register is set to the value C0DE_C0DEh, the value in this field is copied to the following register fields in the flash controller: • MSIZE[MAXADDR0] = FMUPARM4[7:0] Note: If FFh, physical size is stored.	0

45.1.6.2.44 Flash TME control (FMUPARM5)

Table 383. Flash TME control - offset: 1650h)

Bit	Name	Access	Description and value	Reset value
31:0	FIELD	R/W	When FMUPARM0_0 register is set to the value C0DE_C0DEh, the value in this field is copied to the following register fields in the flash controller: • FTEST[TMECTL] = FMUPARM5[0]	0

45.1.6.2.45 SoC trim settings enable (SOCTRIMEN)

Table 384. SoC trim settings enable - offset: 17E0h)

Bit	Name	Access	Description and value
31:0	FIELD	R	When this register contains the value C0DE_C0DEh, the values in the SOCTRIM fields is copied to the corresponding registers.

45.1.6.2.46 SoC trim ECC decoder enable (SOCTRIMECC)

Table 385. SoC trim ECC decoder enable - offset: 17F0h)

Bit	Name	Access	Description and value
31:0	FIELD	R	When this register contains the value C0DE_C0DEh, the ECC decoder is enabled while the SOCTRIM fields is copied to the corresponding registers.

45.1.6.2.47 SOCTRIM0 FRO32K (GPO0_0)

Table 386. SOCTRIM0 FRO32K - offset: 1800h)

Bit	Name	Access	Description and value
31	DCDC	R	DC-to-DC
			0 DCDC enabled (supported modes are indicated by PMIC_ONLY bit)
			1 DCDC disabled in both buck and boost mode
30	PMIC_ONLY	R	DC-to-DC
			0 DCDC configurable in both buck and boost mode
			1 DCDC boost mode is disabled
29:16	reserved	-	reserved Read value is undefined, only zero should be written.
15:7	FRO32K_CAPCAL	R	CAPCAL trim value for the 32.768 kHz FRO
6:4	FRO32K_PTAT	R	PTAT trim value for the 32.768 kHz FRO
3:1	FRO32K_NTAT	R	NTAT trim value for the 32.768 kHz FRO
0	reserved	-	reserved Read value is undefined, only zero should be written.

45.1.6.2.48 SOCTRIM0 flash speed (GPO0_2)

Table 387. SOCTRIM0 flash speed - offset: 1808h)

Bit	Name	Access	Description and value
31:4	reserved	-	reserved Read value is undefined, only zero must be written.
3:2	FLASH_CTRL_OPMODE	R	system speed codeFlash controller speed
			00 FRO 12 MHz
			01 reserved
			10 reserved
			11 FRO 32 MHz
1:0	SYSTEM_SPEED_CODE	R	system speed code
			00 FRO 12 MHz
			01 FRO 16 MHz
			10 FRO 24 MHz
			11 FRO 32 MHz

45.1.6.2.49 SOCTRIM1 device ID (GPO1_0)

Table 388. SOCTRIM1 device ID - offset: 1810h)

Bit	Name	Access	Description and value
31:28	METAL_REVISION	R	metal revision ID[3:0]
27:24	ROM_REVISION_MINOR	R	ROM revision minor [3:0]
23:20	reserved	-	reserved Read value is undefined, only zero must be written.
19:16	CM33_SECURITY_EXTENSION_DISABLE	R	CM33_SECURITY_EXTENSION_DISABLE[7:4]
			1010 CM33 security extension is disabled
			other CM33 security extension is enabled
15:12	SRAM_SIZE	R	SRAM_SIZE[3:0]
			0000 128 KB
			0001 96 KB
			0010 64 KB
			0011 48 KB
			0100 32 KB
			other reserved
11	DEVICE_TYPE_SEC	R	security device type
			1 secure family
10:4	PARTCONFIG	R	Device type 0x38 for MCXW236x 0x35 for MCXW235B

Table 388. SOCTRIM1 device ID - offset: 1810h)...continued

Bit	Name	Access	Description and value
3:0	FINAL_TEST_NOT_DONE	R	FINAL_TEST_NOT_DONE[7:4]
			1010 final test not done
			other final test done

45.1.6.2.50 SOCTRIM1 device revision (GPO1_1)

Table 389. SOCTRIM1 device revision - offset: 1814h)

Bit	Name	Access	Description and value
31:8	reserved	R	reserved Read value is undefined, only zero must be written.
7:4	CUSTOMER_REVISION_ID	R	customer revision ID [3:0]
3:0	ROM_PATCH_VERSION	R	ROM patch version [3:0]

45.1.6.2.51 SOCTRIM2 FRO192M (GPO2_0)

Table 390. SOCTRIM2 FRO192M - offset: 1820h)

Bit	Name	Access	Description and value
31:28	CM33_SECURITY_EXTENSION_DISABLE	R	CM33_SECURITY_EXTENSION_DISABLE[3:0]
			1010 CPU0 security extension is disabled
			other CPU0 security extension is enabled
27:24	FLASH_SIZE	R	0100 512 KB
			1000 1024 KB
			other reserved
23:16	FRO192M_DACTRIM	R	FRO192M fine frequency trimming bits
15:14	reserved	-	reserved Read value is undefined, only zero must be written. be written.
13:8	FRO192M_TEMPTRIM	R	FRO192M temperature coefficient trimming bits
7	reserved	-	reserved Read value is undefined, only zero must be written. be written.
6:1	FRO192M_BIASTRIM	R	FRO192M coarse frequency trimming bits
0	reserved	-	reserved Read value is undefined, only zero must be written. be written.

45.1.6.2.52 SOCTRIM3 band gap (GPO3_0)

Table 391. SOCTRIM3 band gap - offset: 1830h)

Bit	Name	Access	Description and value
31:28	FINAL_TEST_NOT_DONE	R	FINAL_TEST_NOT_DONE[3:0]

Table 391. SOCTRIM3 band gap - offset: 1830h)...continued

Bit	Name	Access	Description and value	
			1010	final test not done
			other	final test done
27:25	MODELNUM_EXTENSION	R	model number extension [2:0]	
24:23	reserved	-	reserved	
22:16	FRO1M_FREQ_SEL	R	1 MHz FRO frequency trim	
15:12	BANDGAP_LP_DC	R	band gap LP-DC shift	
11:10	BANDGAP_TC2	R	band gap curve	
9:5	BANDGAP_BIAS	R	band gap IREFTRIM	
4:0	BANDGAP_TC0	R	band gap DC shift	

45.2 SRAM

[Table 392](#) describes the RAM configuration for the MCX W23.

Table 392. RAM configuration

Device type	RAM-X (KB)	RAM0 (KB)	RAM1 (KB)	RAM2 (KB)	Bluetooth Low Energy RAM (KB)
MCXW236x	16	16	32	64	16 + 0.5
MCXW235x	16	16	32	32	16 + 0.5

45.3 External flash

[Table 393](#) describes the external flash memory map the MCX W23.

Table 393. External flash memory map

Memory	Address
SPIFI data	2800_0000h to 2FFF_FFFFh Note: The address space allocated to the SPIFI for direct execution. The area allocated allows maximum 128 MB of SPI flash to be mapped into the CPU memory space. In practice, the usable space is limited to the size of the connected device.

45.4 APB peripherals

[Table 394](#) provides details of the address for APB peripherals. APB peripherals have secure and non-secure access.

Table 394. APB peripherals memory map

Peripheral	APB bridge	Non-secure base address	Secure base address	Reference
system control (SYSCON)	0	4000_0000h	5000_0000h	Section 45.4.1
pin function selection and pin control setup (IOCON)		4000_1000h	5000_1000h	Section 45.4.2
Group GPIO input interrupt (GINT)		4000_2000h	5000_2000h	Section 45.4.3
pin interrupt and pattern match (PINT)		4000_4000h	5000_4000h	Section 45.4.4
secure pin interrupt and pattern match (SECPINT)		4000_5000h	5000_5000h	Section 45.4.4
input multiplexing and frequency measurement (INPUTMUX)		4000_6000h	5000_6000h	Section 45.4.5
standard counter/timer 0 (CTIMER0)		4000_8000h	5000_8000h	Section 45.4.6
standard counter/timer 1 (CTIMER1)		4000_9000h	5000_9000h	Section 45.4.6
windowed watchdog timer (WWDT)		4000_C000h	5000_C000h	Section 45.4.7
multi-rate timer (MRT)		4000_D000h	5000_D000h	Section 45.4.8
micro-tick timer (UTICK)		4000_E000h	5000_E000h	Section 45.4.9
analog control (ANACTRL)		4001_3000h	5001_3000h	Section 45.4.10
power management controller (PMC)	1	4002_0000h	5002_0000h	Section 45.4.11
system control (SYSCTL)	1	4002_3000h	5002_3000h	Section 45.4.12
standard counter/timer 2 (CTIMER2)		4002_8000h	5002_8000h	Section 45.4.6
standard counter/timer 3 (CTIMER3)		4002_9000h	5002_9000h	Section 45.4.6
standard counter/timer 4 (CTIMER4)		4002_A000h	5002_A000h	Section 45.4.6
RTC and wake-up timer (RTC)		4002_C000h	5002_C000h	Section 45.4.13
OS event timer (OSTIMER)		4002_D000h	5002_D000h	Section 45.4.14
flash memory unit (FMU)	1	4003_4000h	5003_4000h	Section 45.4.15
dynamic encryption/decryption engine (PRINCE)		4003_5000h	5003_5000h	Section 45.4.16
true random number generator (TRNG)		4003_A000h	5003_A000h	Section 45.4.17
physical unclonable function (PUF)		4003_B000h	5003_B000h	Section 45.4.18
programmable logic unit (PLU)		4003_D000h	5003_D000h	Section 45.4.20
ROM control and patch (ROMCP)		4003_E000h	5003_E000h	Section 45.4.21

45.4.1 System control (SYSCON)

45.4.1.1 SYSCON memory map

Table 395. SYSCON - base address: 4000_0000h

Name	Access	Offset	Description	Width (bits)	Reset value	Section
MEMORYREMAP	R/W	0h	memory remap control	32		Section 45.4.1.2.1
AHBMATPRIO	R/W	10h	AHB matrix priority control	32		Section 45.4.1.2.2
BUFFERINGAHB2VPB0	R/W	20h	buffering of write accesses on synchronous system configuration APB interface	32		Section 45.4.1.2.3
BUFFERINGAHB2VPB1	R/W	24h	buffering of write accesses on synchronous system configuration APB interface	32		Section 45.4.1.2.4
CPU0STCKCAL	R/W	38h	system tick calibration for secure part of CPU0	32		Section 45.4.1.2.5
CPU0NSTCKCAL	R/W	3Ch	system tick calibration for non-secure part of CPU0	32		Section 45.4.1.2.6
NMISRC	R/W	48h	NMI source select	32		Section 45.4.1.2.7
PRESETCTRL0	R/W	100h	peripheral reset control 0	32		Section 45.4.1.2.8
PRESETCTRL1	R/W	104h	peripheral reset control 1	32		Section 45.4.1.2.9
PRESETCTRL2	R/W	108h	peripheral reset control 2	32		Section 45.4.1.2.10
PRESETCTRLSET0 - PRESETCTRLSET2	R/W	120h - 128h	peripheral reset control set	32	0000_0000h	Section 45.4.1.2.11
PRESETCTRLCLR0 - PRESETCTRLCLR2	R/W	140h - 148h	peripheral reset control clear	32	0000_0000h	Section 45.4.1.2.12
SWR_RESET	W	160h	software reset	32	0000_0000h	Section 45.4.1.2.13
AHBCLKCTRL0	R/W	200h	AHB clock control 0	32		Section 45.4.1.2.14
AHBCLKCTRL1	R/W	204h	AHB clock control 1	32		Section 45.4.1.2.15
AHBCLKCTRL2	R/W	208h	AHB clock control 2	32		Section 45.4.1.2.16
AHBCLKCTRLSET0 - AHBCLKCTRLSET2	R/W	220h - 228h	AHB clock control set	32	0000_0000h	Section 45.4.1.2.17
AHBCLKCTRLCLR0 - AHBCLKCTRLCLR2	R/W	240h - 248h	AHB clock control clear	32	0000_0000h	Section 45.4.1.2.18
SYSTICKCLKSEL0	R/W	260h	system tick timer for CPU0 source select	32		Section 45.4.1.2.19
TRACECLKSEL	R/W	268h	trace clock source select	32		Section 45.4.1.2.20
CTIMERCLKSEL0 - CTIMERCLKSEL4	R/W	26Ch - 27Ch	CTimer clock source select	32		Section 45.4.1.2.21
MAINCLKSELA	R/W	280h	main clock A source select	32		Section 45.4.1.2.22

Table 395. SYSCON - base address: 4000_0000h...continued

Name	Access	Offset	Description	Width (bits)	Reset value	Section
MAINCLKSELB	R/W	284h	main clock B source select	32		Section 45.4.1.2.23
CLKOUTSEL	R/W	288h	CLKOUT clock source select	32		Section 45.4.1.2.24
FCCLKSEL0	R/W	2B0h	flexcomm interface 0 clock source select for fractional rate divider	32		Section 45.4.1.2.25
FCCLKSEL1	R/W	2B4h	flexcomm interface 1 clock source select for fractional rate divider	32		Section 45.4.1.2.26
FCCLKSEL2	R/W	2B8h	flexcomm interface 2 clock source select for fractional rate divider	32		Section 45.4.1.2.27
SPIFICKSEL	R/W	2D0h	QuadSPI flash interface clock source select	32		Section 45.4.1.2.28
SCTCLKSEL	R/W	2F0h	SCTimer/PWM clock source select	32		Section 45.4.1.2.29
SYSTICKCLKDIV0	R/W	300h	system tick timer divider for CPU0	32		Section 45.4.1.2.30
TRACECLKDIV	R/W	308h	TRACE clock divider	32		Section 45.4.1.2.31
FLEXFRG0CTRL	R/W	320h	fractional rate divider for flexcomm 0	32		Section 45.4.1.2.32
FLEXFRG1CTRL	R/W	324h	fractional rate divider for flexcomm 1	32		Section 45.4.1.2.33
FLEXFRG2CTRL	R/W	328h	fractional rate divider for flexcomm 2	32		Section 45.4.1.2.34
AHBCLKDIV	R/W	380h	system clock divider	32		Section 45.4.1.2.35
CLKOUTDIV	R/W	384h	CLKOUT clock divider	32		Section 45.4.1.2.36
FROHFDIV	R/W	388h	FRO_HF (32 MHz) clock divider	32		Section 45.4.1.2.37
WDTCLKDIV	R/W	38Ch	WDT clock divider	32		Section 45.4.1.2.38
SCTCLKDIV	R/W	3B4h	SCT/PWM clock divider	32		Section 45.4.1.2.39
SPIFICKDIV	R/W	3D0h	QuadSPI flash interface clock divider	32		Section 45.4.1.2.40
CLOCKGENUPDATEL OCKOUT	R/W	3FCh	control clock configuration registers access (like xxxDIV, xxxSEL)	32	0000_0000h	Section 45.4.1.2.41
ROMCR	R/W	404h	ROM access configuration	32		Section 45.4.1.2.42
EZHINT	R/W	414h	EZH interrupt hijack	32		Section 45.4.1.2.43
FLASHREMAP_SIZE	R/W	440h	flash remap size	32	0000_0000h	Section 45.4.1.2.44
FLASHREMAP_SIZE_DP	R/W	444h	flash remap size duplicate	32	0000_0000h	Section 45.4.1.2.45

Table 395. SYSCON - base address: 4000_0000h...continued

Name	Access	Offset	Description	Width (bits)	Reset value	Section
FLASHREMAP_OFFSET	R/W	448h	flash remap offset	32	0000_0000h	Section 45.4.1.2.46
FLASHREMAP_OFFSET_DP	R/W	44Ch	flash remap offset duplicate	32	0000_0000h	Section 45.4.1.2.47
FLASHBANKENABLE	R/W	450h	flash banks control	32		Section 45.4.1.2.48
FLASHREMAP_LOCK	R/W	45Ch	flash remap lock	32	C33C_A55Ah	Section 45.4.1.2.49
CASPER_CTRL	R/W	470h	CASPER control	32		Section 45.4.1.2.50
BOOT_LOCKOUT_ADDR	R/W	480h	boot lockout address offset	32		Section 45.4.1.2.51
BOOT_LOCKOUT_ADDR_DP	R/W	484h	boot lockout address offset duplicate	32		Section 45.4.1.2.52
BOOT_LOCKOUT	R/W	488h	boot lockout	32	3CC3_5AA5h	Section 45.4.1.2.53
STARTER0	W	680h	start logic wake-up enable 0 register	32		Section 45.4.1.2.54
STARTER1	W	684h	start logic wake-up enable 1 register	32		Section 45.4.1.2.55
STARTERSET0	W	6A0h	start logic wake-up enable set 0 register	32		Section 45.4.1.2.56
STARTERSET1	W	6A4h	start logic wake-up enable set 1 register	32		Section 45.4.1.2.57
STARTERCLR0	W	6C0h	start logic wake-up enable clear 0 register	32		Section 45.4.1.2.58
STARTERCLR1	W	6C4h	start logic wake-up enable clear 1 register	32		Section 45.4.1.2.59
FUNCRETENTIONCTRL	R/W	704h	functional retention control register	32		Section 45.4.1.2.60
POWERDOWNSAFETY	R/W	70Ch	power-down control override signal	32		Section 45.4.1.2.61
MAINCLKSAFETY	R/W	710h	main clock safety cycles	32	0000_0000h	Section 45.4.1.2.62
HARDWARESLEEP	R/W	780h	hardware sleep control	32		Section 45.4.1.2.63
CPSTAT	W	80Ch	CPU status	32		Section 45.4.1.2.64
DICE_REG0 - DICE_REG7	R/W	900h - 91Ch	composite device identifier	32	0000_0000h	Section 45.4.1.2.65
BOOT_SEED_REG0 - BOOT_SEED_REG7	R/W	920h - 95Ch	boot seed	32	0000_0000h	Section 45.4.1.2.66
HMAC_REG0 - HMAC_REG7	R/W	940h - 95Ch	HMAC	32	0000_0000h	Section 45.4.1.2.67
BOOT_LOCK	R/W	960h	boot seed lock	32		Section 45.4.1.2.68
CSS_AS_CFG0	W	9D0h	code watchdog status	32		Section 45.4.1.2.69
CLOCK_CTRL	R/W	A18h	clock control	32		Section 45.4.1.2.70
AUTOCLKGATEOVERRIDE	R/W	E04h	control automatic clock gating	32	0000_0000h	Section 45.4.1.2.71

Table 395. SYSCON - base address: 4000_0000h...continued

Name	Access	Offset	Description	Width (bits)	Reset value	Section
GPIOPSYNC	R/W	E08h	GPIO synchronization control	32	0000_FFFFh	Section 45.4.1.2.72
INVERTMAINCLK	R/W	E20h	invert main clock	32		Section 45.4.1.2.73
HASHRESTHWKEY	R/W (once)	F88h	HASH AES key restricted use register	32	0000_0000h	Section 45.4.1.2.74
DEBUG_LOCK_EN	R/W	FA0h	debug lock enable	32		Section 45.4.1.2.75
DEBUG_FEATURES	R/W	FA4h	cortex debug features control	32		Section 45.4.1.2.76
DEBUG_FEATURES_DP	R/W	FA8h	cortex debug features control duplicate	32		Section 45.4.1.2.77
CODESECURITYPROTTEST	W	FB0h	security code to allow test (design for testability) access	32	0000_0000h	Section 45.4.1.2.78
SWD_ACCESS_CPU0	R/W	FB4h	SWD access control	32	0000_0000h	Section 45.4.1.2.79
KEY_BLOCK	W	FBCh	block PUF generated keys	32	3CC3_5AA5h	Section 45.4.1.2.80
DEBUG_AUTH_BEACON	R/W	FC0h	debug authentication BEACON register	32	0000_0000h	Section 45.4.1.2.81
FLASHSIZECFG	R/W	FE0h	flash size configuration	32		Section 45.4.1.2.82
CONFIGLOCKOUT	R/W	FE4h	configuration lockout	32		Section 45.4.1.2.83
RAMSIZECFG	R/W	FE8h	RAM size configuration	32		Section 45.4.1.2.84
PERIPHENCFG	R/W	FECh	peripheral enable configuration	32		Section 45.4.1.2.85
DEVICE_ID0	R/W	FF8h	device ID	32		Section 45.4.1.2.86
DIEID	R	FFCh	chip revision ID and number	32		Section 45.4.1.2.87

45.4.1.2 Register descriptions

45.4.1.2.1 Memory remap control register (MEMORYREMAP)

Table 396. Memory remap control register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1:0	MAP	R/W	location of the vector table	0
			00b vector table in ROM	
			01b vector table in RAM	
			1xb vector table in flash	

45.4.1.2.2 AHB matrix priority control register (AHBMATPRIO)

If several masters attempt to access the same matrix slave port at the same time, the multilayer AHB matrix arbitrates between them. If the value in this register is changed, improper settings can seriously reduce 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 397. AHB matrix priority control register - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:18	reserved	-	reserved Read value is undefined, only zero must be written.	-
17:16	PRI_HASH_AES	R/W	HASH_AES	0
15:14	PRI_EZH_B_I	R/W	EZH B instruction bus	0
13:12	PRI_EZH_B_D	R/W	EZH B data bus	0
11:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
15:8	reserved	-	reserved	-
7:6	PRI_SDMA1	R/W	DMA1 controller priority	0
5:4	PRI_SDMA0	R/W	DMA0 controller priority	0
3:2	PRI_CPU0_SBUS	R/W	Arm CM33 S-AHB bus	0
1:0	PRI_CPU0_CBUS	R/W	Arm CM33 C-AHB bus	0

45.4.1.2.3 Buffering of write accesses on synchronous system configuration APB interface (BUFFERINGAHB2VPB0)

The buffering of write accesses on the internal APB interface adds delay on the bus. If timing issues occur when the IC is used outside its running condition ranges, this buffering can help to keep the interface functional. Do not use this register (RESERVED). Keep the default values.

Table 398. Buffering of write accesses on synchronous system configuration APB interface - Offset: 20h

Bit	Name	Access	Description and value	Reset value
31	PROBXVC	R/W	enable buffering of write accesses on prob IS (XVC) APB interface	0
			0b disable buffering	
			1b enable buffering	
30	PROBSYNC	R/W	enable buffering of write accesses on prob IS (XVC) APB interface	0
			0b disable buffering	
			1b enable buffering	
29	EZH	R/W	enable buffering of write accesses on EZH APB interface	0
			0b disable bufferingreserved	
			1b enable buffering	
28	reserved	-	reserved Read value is undefined, only zero must be written.	-
27	SCTIUP	R/W	enable buffering of write accesses on SCTIUP APB interface	0
			0b disable buffering	
			1b enable buffering	
26:20	reserved	-	reserved Read value is undefined, only zero must be written.	-
19	ANACTRL	R/W	enable buffering of write accesses on analog control APB interface	0
			0b disable buffering	
			1b enable buffering	
18:15	reserved	-	reserved Read value is undefined, only zero must be written.	-
14	UTICK	R/W	enable buffering of write accesses on micro Tick APB interface	0
			0b disable buffering	
			1b enable buffering	
13	MRT	R/W	enable buffering of write access to multirate timer interface	0
			0b disable buffering	
			1b enable buffering	

Table 398. Buffering of write accesses on synchronous system configuration APB interface - Offset: 20h...continued

Bit	Name	Access	Description and value	Reset value
12	WWDT	R/W	enable buffering of write accesses on watchdog timer APB interface	0
			0b disable buffering	
			1b enable buffering	
11:10	reserved	-	reserved Read value is undefined, only zero must be written.	-
9	CT32B1	R/W	enable buffering of write accesses on counter/timer1 APB interface	0
			0b disable buffering	
			1b enable buffering	
8	CT32B0	R/W	enable buffering of write accesses on counter/timer0 APB interface	0
			0b disable buffering	
			1b enable buffering	
7	reserved	-	reserved Read value is undefined, only zero must be written.	-
6	INPUTMUX	R/W	enable buffering of write accesses on peripheral input mux APB interface	0
			0b disable buffering	
			1b enable buffering	
5	SECGPIoint	R/W	enable buffering of write accesses on secure GPIO Int APB interface	0
			0b disable buffering	
			1b enable buffering	
4	GPIoint	R/W	enable buffering of write accesses on GPIO Int APB interface	0
			0b disable buffering	
			1b enable buffering	
3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2	GPIOGLO BALINT0	R/W	enable buffering of write accesses on GPIO global interrupt APB interface	0
			0b disable buffering	
			1b enable buffering	

Table 398. Buffering of write accesses on synchronous system configuration APB interface - Offset: 20h...continued

Bit	Name	Access	Description and value	Reset value
1	IOCON	R/W	enable buffering of write accesses on IO configuration APB interface	0
			0b disable buffering	
			1b enable buffering	
0	SYSCON	R/W	enable buffering of write accesses on synchronous system configuration APB interface	0
			0b disable buffering	
			1b enable buffering	

45.4.1.2.4 Buffering of write accesses on synchronous system configuration APB interface (BUFFERINGAHB2VPB1)

The buffering of write accesses on the internal APB interface adds delay on the bus. If timing issues occur when the IC is used outside its running condition ranges, this buffering can help to keep the interface functional. Recommendation is not to use this register (RESERVED), and to keep default values.

Table 399. Buffering of write accesses on synchronous system configuration APB interface - Offset: 24h

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved Read value is undefined, only zero must be written.	-
30	CODEPATCH	R/W	enable buffering of write accesses on code patch unit APB interface	0
			0b disable buffering	
			1b enable buffering	
29	PLU_LUT	R/W	enable buffering of write accesses on PLU LUT interface	0
			0b disable buffering	
			1b enable buffering	
28	reserved	-	reserved Read value is undefined, only zero must be written.	-
27	PUF	R/W	enable buffering of write accesses on PUFF interface	0
			0b disable buffering	
			1b enable buffering	
26	RNG	R/W	enable buffering of write accesses on random number generator APB interface	0
			0b disable buffering	
			1b enable buffering	
25:22	reserved	-	reserved Read value is undefined, only zero must be written.	-

Table 399. Buffering of write accesses on synchronous system configuration APB interface - Offset: 24h...continued

Bit	Name	Access	Description and value		Reset value
21	PRINCE	R/W	enable buffering of write accesses on prince APB interface		0
			0b	disable buffering	
			1b	enable buffering	
20	FLASHCTRL	-	enable buffering of write accesses on flash controller APB interface		0
			0b	disable buffering	
			1b	enable buffering	
19:14	reserved	-	reserved		-
13	OSEVENT	R/W	enable buffering of write accesses on OS event timer APB interface		0
			0b	disable buffering	
			1b	enable buffering	
12	RTC	R/W	enable buffering of write accesses on RTC APB interface		0
			0b	disable buffering	
			1b	enable buffering	
11	reserved	-	reserved		-
10	CT32B4	R/W	enable buffering of write accesses on counter/timer4 APB interface		0
			0b	disable buffering	
			1b	enable buffering	
9	CT32B3	R/W	enable buffering of write accesses on counter/timer3 APB interface		0
			0b	disable buffering	
			1b	enable buffering	
8	CT32B2	R/W	enable buffering of write accesses on counter/timer2 APB interface		0
			0b	disable buffering	
			1b	enable buffering	
7:4	reserved	-	reserved Read value is undefined, only zero must be written.		-
3	SYSCTL	R/W	enable buffering of write accesses on system control APB interface		0
			0b	disable buffering	
			1b	enable buffering	
2:1	reserved	-	reserved Read value is undefined, only zero must be written.		-
0	PMC	R/W	enable buffering of write accesses on power management controller APB interface		0
			0b	disable buffering	
			1b	enable buffering	

45.4.1.2.5 System tick calibration for secure part of CPU0 (CPU0STCKCAL)

Table 400. System tick calibration for secure part of CPU0 - Offset: 38h

Bit	Name	Access	Description and value	Reset value
31:26	reserved	-	reserved Read value is undefined, only zero must be written.	-
25	NOREF	R/W	indicates if the device provides a reference clock to the processor	0
			0 reference clock provided	
			1 no reference clock provided	
24	SKEW	-	initial value for the systick timer	0
23:0	TENMS	R/W	reload value for 10 ms (100 Hz) timing; subject to system clock skew errors. If the value reads as zero, the calibration value is not known.	0

45.4.1.2.6 System tick calibration for non-secure part of CPU0 (CPU0NSTCKCAL)

Table 401. System tick calibration for non-secure part of CPU0 - Offset: 3Ch

Bit	Name	Access	Description and value	Reset value
31:26	reserved	-	reserved Read value is undefined, only zero must be written.	-
25	NOREF	R/W	initial value for the systick timer	0
24	SKEW	R/W	indicates if the TENMS value is exact	0
			0 TENMS value is exact	
			1 TENMS value is inexact, or not given	
23:0	TENMS	R/W	reload value for 10 ms (100 Hz) timing; subject to system clock skew errors. If the value reads as zero, the calibration value is not known.	0

45.4.1.2.7 NMI source select (NMISRC)

Table 402. NMI source select - Offset: 48h

Bit	Name	Access	Description and value	Reset value
31	NMIENCPU0	R/W	write a 1 to this bit to enable the non-maskable interrupt (NMI) source selected by IRQCPU0.	0
30:6	reserved	-	reserved Read value is undefined, only zero must be written.	-
5:0	IRQCPU0	R/W	The IRQ number of the interrupt that acts as the non-maskable interrupt (NMI) for the CPU0, if enabled by NMIENCPU0.	0

45.4.1.2.8 Peripheral reset control 0 (PRESETCTRL0)

Table 403. Peripheral reset control 0 - Offset: 100h

Bit	Name	Access	Description and value	Reset value
31:24	reserved	-	reserved Read value is undefined, only zero must be written.	-
23	RTC_RST	R/W	real time clock (RTC) reset control	0
			0b bloc is not reset	
			1b bloc is reset	
22	WWDT_RST	R/W	watchdog timer reset control	0
			0b bloc is not reset	
			1b bloc is reset	
21	CRCGEN_RST	R/W	CRCGEN reset control	0
			0b bloc is not reset	
			1b bloc is reset	
20	DMA0_RST	R/W	DMA0 reset control	0
			0b bloc is not reset	
			1b bloc is reset	
19	GINT_RST	R/W	group interrupt (GINT) reset control	0
			0b bloc is not reset	
			1b bloc is reset	
18	PINT_RST	R/W	pin interrupt (PINT) reset control	0
			0b bloc is not reset	
			1b bloc is reset	
17:15	reserved	-	reserved Read value is undefined, only zero must be written.	-
14	GPIO0_RST	R/W	GPIO0 reset control	0
			0b bloc is not reset	
			1b bloc is reset	
13	IOCON_RST	R/W	I/O controller reset control	0
			0b bloc is not reset	
			1b bloc is reset	
12	reserved	-	reserved Read value is undefined, only zero must be written.	-
11	MUX_RST	R/W	input mux reset control	0
			0b bloc is not reset	
			1b bloc is reset	

Table 403. Peripheral reset control 0 - Offset: 100h...continued

Bit	Name	Access	Description and value	Reset value
10:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8	FMC_RST	R/W	FMC controller reset control	0
			0b bloc is not reset	
			1b bloc is reset	
7	FLASH_RST	R/W	flash controller reset control	0
			0b bloc is not reset	
			1b bloc is reset	
6:5	reserved	-	reserved Read value is undefined, only zero must be written.	-
4	SRAM_CTRL2_RST	R/W	SRAM controller 2 reset control	0
			0b bloc is not reset	
			1b bloc is reset	
3	SRAM_CTRL1_RST	R/W	SRAM controller 1 reset control	0
			0b bloc is not reset	
			1b bloc is reset	
2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	ROM_RST	R/W	ROM reset control	0
			0b bloc is not reset	
			1b bloc is reset	
0	reserved	-	reserved Read value is undefined, only zero must be written. This bit must only be used in production when accessed via JTAG. It controls the reset input of the ARM CPU block. When 1b is written, the ARM CPU block is in reset. Write 0b to get ARM CPU block out of reset	-

45.4.1.2.9 Peripheral reset control 1 (PRESETCTRL1)

Table 404. Peripheral reset control 1 - Offset: 104h

Bit	Name	Access	Description and value	Reset value
31	EZHB_RST	R/W	EZH B reset control	0
			0b bloc is not reset	
			1b bloc is reset	
30	EZHA_RST	R/W	EZH A reset control	0
			0b bloc is not reset	
			1b bloc is reset	
31:30	reserved	-	reserved	-
29:28	reserved	-	reserved Read value is undefined, only zero must be written.	-
27	TIMER1_RST	R/W	timer 1 reset control	0
			0b bloc is not reset	
			1b bloc is reset	
26	TIMER0_RST	R/W	timer 0 reset control	0
			0b bloc is not reset	
			1b bloc is reset	
25:23	reserved	-	reserved Read value is undefined, only zero must be written.	-
22	TIMER2_RST	R/W	timer 2 reset control	0
			0b bloc is not reset	
			1b bloc is reset	
21:14	reserved	-	reserved Read value is undefined, only zero must be written.	-
13	FC2_RST	R/W	FC2 reset control	0
			0b bloc is not reset	
			1b bloc is reset	
12	FC1_RST	R/W	FC1 reset control	0
			0b bloc is not reset	
			1b bloc is reset	
11	FC0_RST	R/W	FC0 reset control	0
			0b bloc is not reset	
			1b bloc is reset	
10	UTICK_RST	R/W	UTICK reset control	0
			0b bloc is not reset	
			1b bloc is reset	
9:7	reserved	-	reserved	-

Table 404. Peripheral reset control 1 - Offset: 104h...continued

Bit	Name	Access	Description and value	Reset value
			Read value is undefined, only zero must be written.	
6	SCTIPU_RST	R/W	SCTIPU reset control	0
			0b bloc is not reset	
			1b bloc is reset	
5:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
6:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2	SCT_RST	R/W	SCT reset control	0
			0b bloc is not reset	
			1b bloc is reset	
1	OSTIMER_RST	R/W	OS event timer reset control	0
			0b bloc is not reset	
			1b bloc is reset	
0	MRT_RST	R/W	MRT reset control	0
			0b bloc is not reset	
			1b bloc is reset	

45.4.1.2.10 Peripheral reset control 2 (PRESETCTRL2)

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.

Note: 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 405. Peripheral reset control 2 - Offset: 108h

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved Read value is undefined, only zero must be written.	-
30	GPIO_SEC_INT_RST	R/W	GPIO secure int reset control	0
			0b bloc is not reset	
			1b bloc is reset	
29	GPIO_SEC_RST	R/W	GPIO secure reset control	0
			0b bloc is not reset	
			1b bloc is reset	
28	SPIFI_RST	R/W	QuadSPI Flash Interface reset control	0
			0b bloc is not reset	

Table 405. Peripheral reset control 2 - Offset: 108h...continued

Bit	Name	Access	Description and value		Reset value
			1b	bloc is reset	
27	ANALOG_CTRL_RST	R/W	analog control reset control		0
			0b	bloc is not reset	
			1b	bloc is reset	
26:25	reserved	-		reserved Read value is undefined, only zero must be written.	-
24	CASPER_RST	R/W	Casper reset control		0
			0b	bloc is not reset	
			1b	bloc is reset	
23	PUF_RST	R/W	PUF reset control reset control		0
			0b	bloc is not reset	
			1b	bloc is reset	
22	TIMER4_RST	R/W	timer 4 reset control		0
			0b	bloc is not reset	
			1b	bloc is reset	
21	TIMER3_RST	R/W	timer 3 reset control		0
			0b	bloc is not reset	
			1b	bloc is reset	
20	PLULUT_RST	R/W	PLU LUT reset control		0
			0b	bloc is not reset	
			1b	bloc is reset	
19	reserved	-		reserved Read value is undefined, only zero must be written.	-
18	HASH_AES_RST	R/W	HASH_AES reset control		0
			0b	bloc is not reset	
			1b	bloc is reset	
17:16	reserved	-		reserved Read value is undefined, only zero must be written.	-
15	SYSCTL_RST	R/W	SYSCTL block reset		0
			0b	bloc is not reset	
			1b	bloc is reset	
14	reserved	-		reserved Read value is undefined, only zero must be written.	-

Table 405. Peripheral reset control 2 - Offset: 108h...continued

Bit	Name	Access	Description and value	Reset value
13	RNG_RST	R/W	RNG reset control	0
			0b bloc is not reset	
			1b bloc is reset	
12	reserved	-	reserved Read value is undefined, only zero must be written.	-
11	CWT_RST	R/W	code watchdog reset control	0
			0b bloc is not reset	
			1b bloc is reset	
10:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8	FREQME_RST	R/W	frequency meter reset control	0
			0b bloc is not reset	
			1b bloc is reset	
7	BLE_RADIO_RST	R/W	Bluetooth Low Energy radio reset control	0
			0b bloc is not reset	
			1b bloc is reset	
6	BLE_RAM	R/W	Bluetooth Low Energy packet RAM reset control	0
			0b bloc is not reset	
			1b bloc is reset	
5:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	DMA1_RST	R/W	DMA1 reset control	0
			0b bloc is not reset	
			1b bloc is reset	
0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.4.1.2.11 Peripheral reset control set register (PRESETCTRLSET0 - PRESETCTRLSET2)

Table 406. Offset

Register	Offset
PRESETCTRLSET0	120h
PRESETCTRLSET1	124h
PRESETCTRLSET2	128h

Table 407. Peripheral reset control set register

Bit	Name	Access	Description and value	Reset value
31:0	DATA	R/W	data array value	0

45.4.1.2.12 Peripheral reset control clear register (PRESETCTRLCLR0 - PRESETCTRLCLR2)

Table 408. Offset

Register	Offset
PRESETCTRLCLR0	140h
PRESETCTRLCLR1	144h
PRESETCTRLCLR2	148h

Table 409. Peripheral reset control clear register

Bit	Name	Access	Description and value	Reset value
31:0	DATA	R/W	data array value	0

45.4.1.2.13 Software reset (SWR_RESET)

When writing the value 5A00_0001h to this register, a software_reset is generated.

Table 410. Peripheral reset control clear register

Bit	Name	Access	Description and value	Reset value
31:0	SW_RESET	R/W	write 5A00_0001h to generate a software_reset	0
			0000_0000h: Bloc is not reset.	
			5A00_0001h: Generate a software reset	

45.4.1.2.14 AHB clock control 0 (AHBCLKCTRL0)

Table 411. AHB clock control 0 - Offset: 200h

Bit	Name	Access	Description and value	Reset value
31:24	reserved	-	reserved Read value is undefined, only zero must be written.	-
23	RTC	R/W	enables the clock for the real time clock (RTC)	0
			0b disable clock	
			1b enable clock	
22	WWDT	R/W	enables the clock for the watchdog timer	0
			0b disable clock	
			1b enable clock	
21	CRCGEN	R/W	enables the clock for the CRCGEN	0
			0b disable clock	
			1b enable clock	
20	DMA0	R/W	enables the clock for the DMA0	0
			0b disable clock	
			1b enable clock	
19	GINT	R/W	enables the clock for the group interrupt (GINT)	0
			0b disable clock	
			1b enable clock	
18	PINT	R/W	enables the clock for the pin interrupt (PINT)	0
			0b disable clock	
			1b enable clock	
17:15	reserved	-	reserved Read value is undefined, only zero must be written.	-
14	GPIO0	R/W	enables the clock for the GPIO0	0
			0b disable clock	
			1b enable clock	
13	IOCON	R/W	enables the clock for the I/O controller	0
			0b disable clock	
			1b enable clock	
12	reserved	-	reserved Read value is undefined, only zero must be written.	-
11	MUX	R/W	enables the clock for the input mux	0
			0b disable clock	
			1b enable clock	
10:9	reserved	-	reserved Read value is undefined, only zero must be written.	-

Table 411. AHB clock control 0 - Offset: 200h...continued

Bit	Name	Access	Description and value	Reset value
8	FMC	R/W	enables the clock for the flash controller during erase, program and read	1
			0b disable clock	
			1b enable clock	
7	FLASH	R/W	enables the clock for the flash controller register interface during flash erase and program	1
			0b disable clock	
			1b enable clock	
6:5	reserved	-	reserved Read value is undefined, only zero must be written.	-
4	SRAM_CTRL2	R/W	enables the clock for the SRAM controller 2	0
			0b disable clock	
			1b enable clock	
3	SRAM_CTRL1	R/W	enables the clock for the SRAM controller 1	0
			0b disable clock	
			1b enable clock	
2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	ROM	R/W	enables the clock for the ROM	0
			0b disable clock	
			1b enable clock	
0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.4.1.2.15 AHB clock control 1 (AHBCLKCTRL1)

Table 412. AHB clock control 1 - Offset: 204h

Bit	Name	Access	Description and value	Reset value
31	EZHB	R/W	enables the clock for the EZH b	0
			0b disable clock	
			1b enable clock	
30	EZHA	R/W	enables the clock for the EZH a	0
			0b disable clock	
			1b enable clock	
29:28	reserved	-	reserved Read value is undefined, only zero must be written.	-
31:28	reserved	-	reserved	-
27	TIMER1	R/W	enables the clock for the timer 1	0

Table 412. AHB clock control 1 - Offset: 204h...continued

Bit	Name	Access	Description and value		Reset value
			0b	disable clock	
			1b	enable clock	
26	TIMER0	R/W	enables the clock for the timer 0		0
			0b	disable clock	
			1b	enable clock	
25:23	reserved	R/W	reserved Read value is undefined, only zero must be written.		0
22	TIMER2	R/W	enables the clock for the timer 2		0
			0b	disable clock	
			1b	enable clock	
21:14	reserved	-	reserved Read value is undefined, only zero must be written.		-
13	FC2	R/W	enables the clock for the FC2		0
			0b	disable clock	
			1b	enable clock	
12	FC1	R/W	enables the clock for the FC1		0
			0b	disable clock	
			1b	enable clock	
11	FC0	R/W	enables the clock for the FC0		0
			0b	disable clock	
			1b	enable clock	
10	UTICK	R/W	enables the clock for the UTICK		0
			0b	disable clock	
			1b	enable clock	
9:3	reserved	-	reserved Read value is undefined, only zero must be written.		-
2	SCT	R/W	enables the clock for the SCT		0
			0b	disable clock	
			1b	enable clock	
1	OSTIMER	R/W	enables the clock for the OS event timer		0
			0b	disable clock	
			1b	enable clock	
0	MRT	R/W	enables the clock for the MRT		0
			0b	disable clock	
			1b	enable clock	

45.4.1.2.16 AHB clock control 2 (AHBCLKCTRL2)

Table 413. AHB clock control 2 - Offset: 208h

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved Read value is undefined, only zero must be written.	-
30	GPIO_SEC_INT	R/W	enables the clock for the GPIO secure interrupt	0
			0b disable clock	
			1b enable clock	
29	GPIO_SEC	R/W	enables the clock for the GPIO secure	0
			0b disable clock	
			1b enable clock	
28	SPIFI	R/W	enables the clock for the QuadSPI flash interface	0
			0b disable clock	
			1b enable clock	
27	ANALOG_CTRL	R/W	enables the clock for the analog control	0
			0b disable clock	
			1b enable clock	
26:25	reserved	R/W	reserved Read value is undefined, only zero must be written.	-
24	CASPER	R/W	enables the clock for the Casper	0
			0b disable clock	
			1b enable clock	
23	PUF	R/W	enables the clock for the PUF reset control	0
			0b disable clock	
			1b enable clock	
22	TIMER4	R/W	enables the clock for the timer 4	0
			0b disable clock	
			1b enable clock	
21	TIMER3	R/W	enables the clock for the timer 3	0
			0b disable clock	
			1b enable clock	
20	PLULUT	R/W	enables the clock for the PLU LUT	0
			0b disable clock	
			1b enable clock	
19	reserved	-	reserved Read value is undefined, only zero must be written.	-

Table 413. AHB clock control 2 - Offset: 208h...continued

Bit	Name	Access	Description and value	Reset value
18	HASH_AES	R/W	enables the clock for the HASH_AES	0
			0b disable clock	
			1b enable clock	
17:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15	SYSCTL	R/W	enables the clock for the SYSCTL	0
			0b disable clock	
			1b enable clock	
14	reserved	-	reserved Read value is undefined, only zero must be written.	-
13	RNG	R/W	enables the clock for the RNG	0
			0b disable clock	
			1b enable clock	
12:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8	FREQME	R/W	enables the clock for the frequency measurement unit	0
			0b disable clock	
			1b enable clock	
7	BLE_RADIO	R/W	enables the clock for the Bluetooth Low Energy radio	0
			0b disable clock	
			1b enable clock	
6	BLE_RAM	R/W	enables the clock for the Bluetooth Low Energy packet RAM controller	0
			0b disable clock	
			1b enable clock	
5:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	DMA1	R/W	enables the clock for the DMA1	0
			0b disable clock	
			1b enable clock	
0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.4.1.2.17 AHB clock control set register (AHBCLKCTRLSET0 - AHBCLKCTRLSET2)

Table 414. Offset

Register	Offset
AHBCLKCTRSET0	220h
AHBCLKCTRLSET1	224h
AHBCLKCTRLSET2	228h

Table 415. Peripheral reset control set register

Bit	Name	Access	Description and value	Reset value
31:0	DATA	R/W	data array value	0

45.4.1.2.18 AHB clock control clear register (AHBCLKCTRLCLR0 - AHBCLKCTRLCLR2)

Table 416. Offset

Register	Offset
AHBCLKCTRCLR0	240h
AHBCLKCTRLCLR1	244h
AHBCLKCTRLCLR2	248h

Table 417. Peripheral reset control set register

Bit	Name	Access	Description and value	Reset value
31:0	DATA	R/W	data array value	0

45.4.1.2.19 System tick timer source select (SYSTICKCLKSEL0)

Table 418. System tick timer source select - Offset: 260h

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2:0	SEL	R/W	system tick timer for Arm CM33 source select	1
			000b system tick 0 divided clock	
			001b FRO 1 MHz clock	
			010b oscillator 32.768 kHz clock	
			other no clock	

45.4.1.2.20 Trace clock source select (TRACECLKSEL)

Table 419. Trace clock source select - Offset: 268h

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2:0	SEL	R/W	trace clock source select 000b trace divided clock 001b FRO 1 MHz clock 010b oscillator 32.768 kHz clock other no clock	1

45.4.1.2.21 CTimer 0 - 4 clock source select (CTIMERCLKSELa)

For a = 0 to 4

Table 420. CTimer 0 - 4 clock source select - Offset: 26Ch + (a × 4)

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2:0	SEL	R/W	CTimer 0 - 4 clock source select 000b main clock 001b no clock 010b no clock 011b FRO 32 MHz clock 100b FRO 1 MHz clock 101b FRO 24 MHz clock 110b oscillator 32.768 kHz clock 111b no clock	1

45.4.1.2.22 Main clock A source select (MAINCLKSELA)

Table 421. Main clock A source select - Offset: 280h)

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1:0	SEL	R/W	Main clock A source select 000b FRO 12 MHz clock 001b CLKIN clock 010b FRO 1 MHz clock 011b FRO 32 MHz clock	0

45.4.1.2.23 Main clock B source select (MAINCLKSELB)

Table 422. Main clock B source select - Offset: 284h)

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1:0	SEL	R/W	Main clock B source select 000b CLKIN clock 001b FRO 24 MHz clock 010b FRO 32.768 kHz oscillator clock 011b FRO 32.768 kHz oscillator clock	0

45.4.1.2.24 CLKOUT clock source select (CLKOUTSEL)

Table 423. CLKOUT clock source select - Offset: 288h)

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2:0	SEL	R/W	CLKOUT clock source select 000b main clock 001b no clock 010b CLKIN clock 011b FRO 32 MHz clock 100b FRO 1 MHz clock 101b FRO 24 MHz 110b 32.768 kHz oscillator clock other no clock	1

45.4.1.2.25 Flexcomm interface 0 clock source select for fractional rate divider (FCCLKSEL0)

Table 424. Flexcomm interface 0 clock source select for fractional rate divider - Offset: 2B0h)

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2:0	SEL	R/W	Flexcomm interface 0 clock source select for fractional rate divider 000b main clock 001b no clock 010b FRO 12 MHz clock 011b divided FRO 32 MHz clock 100b FRO 1 MHz clock 101b FRO 24 MHz clock 110b 32.768 kHz oscillator clock 111b no clock	1

45.4.1.2.26 Flexcomm interface 1 clock source select for fractional rate divider (FCCLKSEL1)

Table 425. Flexcomm interface 1 clock source select for fractional rate divider - Offset: 2B4h)

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2:0	SEL	R/W	Flexcomm interface 1 clock source select for fractional rate divider 000b main clock 001b no clock 010b FRO 12 MHz clock 011b divided FRO 32 MHz clock 100b FRO 1 MHz clock 101b FRO 24 MHz clock 110b 32.768 kHz oscillator clock 111b no clock	1

45.4.1.2.27 Flexcomm interface 2 clock source select for fractional rate divider (FCCLKSEL2)

Table 426. Flexcomm interface 2 clock source select for fractional rate divider - Offset: 2B8h)

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2:0	SEL	R/W	Flexcomm interface 2 clock source select for fractional rate divider	1
			000b main clock	
			001b no clock	
			010b FRO 12 MHz clock	
			011b divided FRO 32 MHz clock	
			100b FRO 1 MHz clock	
			101b FRO 24 MHz clock	
			110b 32.768 kHz oscillator clock	
			111b no clock	

45.4.1.2.28 QuadSPI flash interface clock source select (SPIFCLKSEL)

Table 427. QuadSPI flash interface clock source select - Offset: 2D0h)

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2:0	SEL	R/W	QuadSPI flash interface clock source select	1
			000b main clock	
			001b FRO 64 MHz clock	
			010b FRO 32 MHz clock	
			011b divided FRO 32 MHz clock	
			1xxb FRO 48 MHz clock	

45.4.1.2.29 SCTimer/PWM clock source select (SCTCLKSEL)

Table 428. SCTimer/PWM clock source select - Offset: 2F0h)

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2:0	SEL	R/W	SCTimer/PWM clock source select	1
			000b main clock	
			001b no clock	
			010b CLKIN clock	
			011b FRO 32 MHz clock	
			1xxb FRO 24 MHz clock	

45.4.1.2.30 System tick timer divider for CPU0 (SYSTICKCLKDIV0)

Table 429. System tick timer divider for CPU0 - Offset: 300h)

Bit	Name	Access	Description and value	Reset value
31	REQFLAG	R/W	divider status flag	0
			0b divider clock is stable	
			1b clock frequency is not stable	
30	HALT	R/W	halts the divider counter	1
			0b divider clock is running	
			1b divider clock is stopped	
29	RESET	R/W	resets the divider counter	0
			0b divider is not reset	
			1b divider is reset	
28:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	DIV	R/W	clock divider value	0

45.4.1.2.31 TRACE clock divider (TRACECLKDIV)

Table 430. TRACE clock divider - Offset: 308h)

Bit	Name	Access	Description and value	Reset value
31	REQFLAG	R/W	divider status flag	0
			0b divider clock is stable	
			1b clock frequency is not stable	
30	HALT	R/W	halts the divider counter	1
			0b divider clock is running	
			1b divider clock is stopped	
29	RESET	R/W	resets the divider counter	0
			0b divider is not reset	
			1b divider is reset	
28:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	DIV	R/W	clock divider value	0

45.4.1.2.32 Fractional rate divider for flexcomm 0 (FLEXFRG0CTRL)

Frequency output = (Frequency of clock selected via FCCLKSEL0) / (1 + MULT / DIV)

Table 431. Fractional rate divider for flexcomm 0 - Offset: 320h)

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15:8	MULT	R/W	numerator of the fractional rate divider	0
7:0	DIV	R/W	denominator of the fractional rate divider	1

45.4.1.2.33 Fractional rate divider for flexcomm 1 (FLEXFRG1CTRL)

Frequency output = (Frequency of clock selected via FCCLKSEL1) / (1 + MULT / DIV)

Table 432. Fractional rate divider for flexcomm 1 - Offset: 324h)

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15:8	MULT	R/W	numerator of the fractional rate divider	0
7:0	DIV	R/W	denominator of the fractional rate divider	1

45.4.1.2.34 Fractional rate divider for flexcomm 2 (FLEXFRG2CTRL)

Frequency output = (Frequency of clock selected via FCCLKSEL2) / (1 + MULT / DIV)

Table 433. Fractional rate divider for flexcomm 2 - Offset: 324h)

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15:8	MULT	R/W	numerator of the fractional rate divider	0
7:0	DIV	R/W	denominator of the fractional rate divider	1

45.4.1.2.35 System clock divider (AHBCLKDIV)

Table 434. System clock divider - Offset: 380h)

Bit	Name	Access	Description and value	Reset value
31	REQFLAG	R/W	divider status flag	0
			0b divider clock is stable	
			1b clock frequency is not stable	
30:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	DIV	R/W	clock divider value	0

45.4.1.2.36 CLKOUT clock divider (CLKOUTDIV)

Table 435. CLKOUT clock divider - Offset: 384h)

Bit	Name	Access	Description and value	Reset value
31	REQFLAG	R/W	divider status flag	0
			0b divider clock is stable	
			1b clock frequency is not stable	
30	HALT	R/W	halts the divider counter	1
			0b divider clock is running	
			1b divider clock is stopped	
29	RESET	R/W	resets the divider counter	0
			0b divider is not reset	
			1b divider is reset	
28:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	DIV	R/W	clock divider value	0

45.4.1.2.37 FRO_HF (32 MHz) clock divider (FROHFDIV)

Table 436. FRO_HF (32 MHz) clock divider - Offset: 388h)

Bit	Name	Access	Description and value	Reset value
31	REQFLAG	R/W	divider status flag	0
			0b divider clock is stable	
			1b clock frequency is not stable	
30:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	DIV	R/W	clock divider value	0

45.4.1.2.38 WDT clock divider (WDTCLKDIV)

Table 437. WDT clock divider - Offset: 38Ch)

Bit	Name	Access	Description and value	Reset value
31	REQFLAG	R/W	divider status flag	0
			0b divider clock is stable	
			1b clock frequency is not stable	
30	HALT	R/W	halts the divider counter	1
			0b divider clock is running	
			1b divider clock is stopped	
29	RESET	R/W	resets the divider counter	0
			0b divider is not reset	
			1b divider is reset	
28:6	reserved	-	reserved Read value is undefined, only zero must be written.	-
5:0	DIV	R/W	clock divider value	0

45.4.1.2.39 SCT/PWM clock divider (SCTCLKDIV)

Table 438. SCT/PWM clock divider - Offset: 3B4h)

Bit	Name	Access	Description and value	Reset value
31	REQFLAG	R/W	divider status flag	0
			0b divider clock is stable	
			1b clock frequency is not stable	
30	HALT	R/W	halts the divider counter	1
			0b divider clock is running	
			1b divider clock is stopped	

Table 438. SCT/PWM clock divider - Offset: 3B4h)...continued

Bit	Name	Access	Description and value	Reset value
29	RESET	R/W	resets the divider counter	0
			0b divider is not reset	
			1b divider is reset	
28:6	reserved	-	reserved Read value is undefined, only zero must be written.	-
5:0	DIV	R/W	clock divider value	0

45.4.1.2.40 QuadSPI flash interface clock divider (SPIFCLKDIV)

Table 439. QuadSPI flash interface clock divider - Offset: 3D0h)

Bit	Name	Access	Description and value	Reset value
31	REQFLAG	R/W	divider status flag	0
			0b divider clock is stable	
			1b clock frequency is not stable	
30	HALT	R/W	halts the divider counter	1
			0b divider clock is running	
			1b divider clock is stopped	
29	RESET	R/W	resets the divider counter	0
			0b divider is not reset	
			1b Divider is reset	
28:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	DIV	R/W	clock divider value	0

45.4.1.2.41 Control clock configuration registers access (CLOCKGENUPDATELOCKOUT)

Table 440. Control clock configuration registers access - Offset: 3FCh)

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved	-
1:0	CLOCKGENUPDATELOCKOUT	R/W	Control clock configuration registers access (for example, xxxDIV, xxxSEL).	0
			0b Clock configuration registers are locked and write to these registers are ignored.	
			1b Clock configuration registers can be modified by writing to it.	

45.4.1.2.42 ROM access configuration register (ROMCR)

Table 441. ROM access configuration register - Offset: 404h)

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved read value is undefined, only zero should be written.	-
0	WAIT	R/W	Extra-wait state between Arm master and ROM access	0
			0b no extra-wait state between Arm master and ROM access	
			1b extra-wait state between Arm master and ROM access	

45.4.1.2.43 EZH interrupt hijack (EZHINT)

Table 442. EZH interrupt hijack - Offset: 414h)

Bit	Name	Access	Description and value	Reset value
31:24	reserved	-	reserved read value is undefined, only zero should be written.	-
23:0	EZHINT	R/W	EZH interrupt hijack	0

45.4.1.2.44 Flash remap size (FLASHREMAP_SIZE)

Table 443. Flash remap size - Offset: 440h)

Bit	Name	Access	Description and value	Reset value
31:0	FLASHREMAP_SIZE	R/W	This 32-bit register contains the size of the image to remap, in bytes. The 12 LSBs are ignored, so the size granularity is 4 KB.	0

45.4.1.2.45 Flash remap size duplicate (FLASHREMAP_SIZE_DP)

The values of FLASHREMAPSIZE and FLASHREMAPSIZE_DP must be set to the same value for their settings to become effective. When the value of FLASHREMAP_SIZE does not equal the value of FLASHREMAP_SIZE_DP, a read of the flash via the FMC results in an abort response.

Table 444. Flash remap size duplicate - Offset: 444h)

Bit	Name	Access	Description and value	Reset value
31:0	FLASHREMAP_SIZE_DP	R/W	duplicate of FLASHREMAP_SIZE for increased security	0

45.4.1.2.46 Flash remap offset (FLASHREMAP_OFFSET)

Table 445. Flash remap offset - Offset: 448h)

Bit	Name	Access	Description and value	Reset value
31:0	FLASHREMAP_OFFSET	R/W	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 4 KB.	0

45.4.1.2.47 Flash remap offset duplicate (FLASHREMAP_OFFSET_DP)

The values of FLASHREMAPOFFSET and FLASHREMAPOFFSET_DP must be set to the same value for their settings to become effective. When the value of FLASHREMAP_OFFSET does not equal the value of FLASHREMAP_OFFSET_DP, a read of the flash via the FMC results in an abort response.

Table 446. Flash remap offset duplicate - Offset: 44Ch)

Bit	Name	Access	Description and value	Reset value
31:0	FLASHREMAP_OFFSET_DP	R/W	duplicate of FLASHREMAP_OFFSET for increased security	0

45.4.1.2.48 Flash banks control (FLASHBANKENABLE)

Table 447. Flash banks control - Offset: 450h)

Bit	Name	Access	Description and value	Reset value
31:20	reserved	-	reserved Read value is undefined, only zero must be written.	-
19:16	IFR1	R/W	IFR1 read access control IFR1 is a flash space containing all trim data for this part.	0
			0000b IFR1 bank is disabled to read	
			1010b IFR1 bank is enabled to read	
15:12	BANK3	R/W	flash BANK3 control BANK3 is a flash memory space for customer field programmable area from address 0x100000 to 0x101FFF.	0
			0000b flash BANK3 checker is enabled All flash pages inside this bank cannot be erased nor programmed.	
			1010b flash BANK3 checker is disabled All flash pages inside this bank can be erased nor programmed.	
11:8	BANK2	R/W	flash BANK2 control BANK2 is a flash memory space for customer field programmable area from address 0x102000 to 0x103FFF.	0
			0000b flash BANK2 checker is enabled All flash pages inside this bank cannot be erased nor programmed.	
			1010b flash BANK2 checker is disabled All flash pages inside this bank can be erased nor programmed.	

Table 447. Flash banks control - Offset: 450h)...continued

Bit	Name	Access	Description and value	Reset value
7:4	BANK1	R/W	flash BANK1 control Bank1 is a flash memory space for customer factory configuration and secret key codes area from address 0x104000 to 0x105FFF.	0
			0000b flash BANK1 checker is enabled All flash pages inside this bank cannot be erased nor programmed.	
			1010b flash BANK1 checker is disabled All flash pages inside this bank can be erased nor programmed.	
3:0	BANK0	R/W	flash BANK0 control BANK0 is a flash memory space for NXP area from address 0x106000 to 0x107FFF.	0
			0000b flash BANK0 checker is enabled All flash pages inside this bank cannot be erased nor programmed.	
			1010b flash BANK0 checker is disabled All flash pages inside this bank can be erased nor programmed.	

45.4.1.2.49 Flash remap lock registers (FLASHREMAP_LOCK)

Table 448. Flash remap lock registers - Offset: 45Ch)

Bit	Name	Access	Description and value	Reset value
31:0	LOCK	R/W	control write access to FLASHREMAP_SIZE and FLASHREMAP_OFFSET registers Any value other than C33C_A55Ah and 3CC3_5AA5h does not modify the state.	C33C-A55Ah
			3CC35AA5h Write access to the registers FLASHREMAP_SIZE0/1/2/3 and FLASHREMAP_OFFSET0/1/2/3 is unlocked.	
			C33CA55Ah Write access to the registers FLASHREMAP_SIZE0/1/2/3 and FLASHREMAP_OFFSET0/1/2/3 is locked.	

45.4.1.2.50 CASPER control register (CASPER_CTRL)

Table 449. CASPER control register - Offset: 470h)

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved Read value is undefined, only zero must be written.	-
0	INTERLEAVE	R/W	control RAM access for RAMX0 and RAMX1	0
			0b RAM access to RAMX0 and RAMX1 is consecutive	
			1b RAM access to RAMX0 and RAMX1 is interleaved	

45.4.1.2.51 Boot lockout address offset (BOOT_LOCKOUT_ADDR)

Table 450. Boot lockout address offset - Offset: 480h)

Bit	Name	Access	Description and value	Reset value
31:15	reserved	-	reserved Read value is undefined, only zero must be written.	-
14:0	ADDR_OFFSET	R/W	address offset (32-bit addressing) Under this value the ROM access is blocked if BOOT_LOCKOUT is LOCK.	0

45.4.1.2.52 Boot lockout address offset duplicate (BOOT_LOCKOUT_ADDR_DP)

Table 451. Boot lockout address offset duplicate - Offset: 484h)

Bit	Name	Access	Description and value	Reset value
31:15	reserved	-	reserved Read value is undefined, only zero must be written.	-
14:0	ADDR_OFFSET	R/W	duplicate of BOOT_LOCKOUT_ADDR for increased security	0

45.4.1.2.53 Boot lockout register (BOOT_LOCKOUT)

Table 452. Boot lockout register - Offset: 488h)

Bit	Name	Access	Description and value	Reset value
31:0	LOCK	R/W	Control write access to BOOT_LOCKOUT_ADDR and BOOT_LOCKOUT_ADDR_DP registers and ROM memory	3CC3_5AA5h
			3CC35AA5h ROM access is allowed, and registers BOOT_LOCKOUT_ADDR and BOOT_LOCKOUT_ADDR_DP are writable.	
			C33CA55Ah ROM access is locked, and registers BOOT_LOCKOUT_ADDR and BOOT_LOCKOUT_ADDR_DP are write protected.	

45.4.1.2.54 Start logic wake-up enable register (STARTER0)

Table 453. Start logic wake-up enable register - Offset: 680h

Bit	Name	Access	Description and value	Reset value
31	WAKEUP	R/W	WAKEUP from deep-sleep interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
30	EZH_ARCH_B0	R/W	EZH_ARCH_B0 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
29	RTC_LITE0	R/W	RTC_LITE0 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
28	BOD2	R/W	BOD2 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
27	BOD1	R/W	BOD1 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
26:25	reserved	-	reserved Read value is undefined, only zero must be written.	-
24	WDT	R/W	windowed watchdog interrupt wake-up	0
23	BLE_SLP_TMR	R/W	Bluetooth Low Energy sleep timer interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
22	BLE_LL	R/W	Bluetooth Low Energy link layer interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	

Table 453. Start logic wake-up enable register - Offset: 680h...continued

Bit	Name	Access	Description and value	Reset value
21:17	reserved	-	reserved Read value is undefined, only zero must be written.	-
16	FLEXINT2	R/W	FLEXINT2 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
15	FLEXINT1	R/W	FLEXINT1 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
14	FLEXINT0	R/W	FLEXINT0 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
13	CTIMER3	R/W	CTIMER3 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
12	SCT0	R/W	SCT0 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
11	CTIMER1	R/W	CTIMER1 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
10	CTIMER0	R/W	CTIMER0 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
9	MRT0	R/W	MRT0 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
8	UTICK0	R/W	UTICK0 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
7	PIO_INT3	R/W	PIO_INT3 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
6	PIO_INT2	R/W	PIO_INT2 interrupt wake-up	0
			Wake-up disabled	
			Wake-up enabled	
5	PIO_INT1	R/W	PIO_INT1 interrupt wake-up	0

Table 453. Start logic wake-up enable register - Offset: 680h...continued

Bit	Name	Access	Description and value		Reset value
			0b	Wake-up disabled	
			1b	Wake-up enabled	
4	PIO_INT0	R/W	PIO_INT0 interrupt wake-up		0
			0b	Wake-up disabled	
			1b	Wake-up enabled	
3	reserved	-	reserved Read value is undefined, only zero must be written.		-
2	GINT0	R/W	GINT0 interrupt wake-up		0
			0b	Wake-up disabled	
			1b	Wake-up enabled	
1	SDMA0	R/W	SDMA0 interrupt wake-up		0
			0b	Wake-up disabled	
			1b	Wake-up enabled	
0	SYS	R/W	system interrupt (WDT, BOD1, or BOD2) wake-up		0
			0b	Wake-up disabled	
			1b	Wake-up enabled	

45.4.1.2.55 Start logic wake-up enable register (STARTER1)

Table 454. Start logic wake-up enable register - Offset: 684h)

Bit	Name	Access	Description and value	Reset value
31	WAKEUPPADS	R/W	WAKEUPPADS interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
30:27	reserved	-	reserved Read value is undefined, only zero must be written.	-
26	SDMA1	R/W	SDMA1 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
25	reserved	-	reserved Read value is undefined, only zero must be written.	-
24	QDDKEY	R/W	QDDKEY interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
23	CASPER	R/W	CASPER interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
22	SHA	R/W	SHA interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
21	SEC_VIO	R/W	Security violation interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
20	PLU	R/W	PLU interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
19	SEC_GPIO_INT01	R/W	SEC_GPIO_INT01 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	
18	SEC_GPIO_INT00	R/W	SEC_GPIO_INT00 interrupt wake-up	0
			0b Wake-up disabled	
			1b Wake-up enabled	

Table 454. Start logic wake-up enable register - Offset: 684h)...continued

Bit	Name	Access	Description and value		Reset value
17	SEC_HYPERVISOR_CALL	R/W	SEC_HYPERVISOR_CALL interrupt wake-up		0
			0b	Wake-up disabled	
			1b	Wake-up enabled	
16:8	reserved	-	reserved Read value is undefined, only zero must be written.		-
7	SPIFI	R/W	SPIFI interrupt wake-up		0
			0b	Wake-up disabled	
			1b	Wake-up enabled	
6	OS_EVENT	R/W	OS_EVENT interrupt wake-up		0
				Wake-up disabled	
				Wake-up enabled	
5	CTIMER4	R/W	CTIMER4 interrupt wake-up		0
			0b	Wake-up disabled	
			1b	Wake-up enabled	
4	CTIMER2	R/W	CTIMER2 interrupt wake-up		0
			0b	Wake-up disabled	
			1b	Wake-up enabled	
3	GPIO_INT07	R/W	GPIO_INT07 interrupt wake-up		0
			0b	Wake-up disabled	
			1b	Wake-up enabled	
2	GPIO_INT06	R/W	GPIO_INT06 interrupt wake-up		0
			0b	Wake-up disabled	
			1b	Wake-up enabled	
1	GPIO_INT05	R/W	GPIO_INT05 interrupt wake-up		0
			0b	Wake-up disabled	
			1b	Wake-up enabled	
0	GPIO_INT04	R/W	GPIO_INT04 interrupt wake-up		0
			0b	Wake-up disabled	
			1b	Wake-up enabled	

45.4.1.2.56 Start logic wake-up enable set 0 register (STARTERSET0)

Table 455. Start logic wake-up enable set 0 register - Offset: 6A0h)

Bit	Name	Access	Description and value	Reset value
31	WAKEUP_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
30	EZH_ARCH_B0_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
29	RTC_LITE0_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
28	BOD2INT_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
27	BOD1INT_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
26:24	reserved	-	reserved Read value is undefined, only zero must be written.	-
23	BLE_SLP_TMR_INT_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
22	BLE_LL_INT_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
21:17	reserved	-	reserved Read value is undefined, only zero must be written.	-
16	FLEXINT2_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
15	FLEXINT1_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
14	FLEXINT0_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
13	CTIMER3_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
12	SCT0_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
11	CTIMER1_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
10	CTIMER0_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
9	MRT0_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
8	UTICK0_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
7	GPIO_INT03_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
6	GPIO_INT02_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0

Table 455. Start logic wake-up enable set 0 register - Offset: 6A0h)...continued

Bit	Name	Access	Description and value	Reset value
5	GPIO_INT01_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
4	GPIO_INT00_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2	GPIO_GLOBALINT0_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
1	SDMA0_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0
0	SYS_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER0 register.	0

45.4.1.2.57 Start logic wake-up enable set 1 register (STARTERSET1)

Table 456. Start logic wake-up enable set 1 register - Offset: 6A4h)

Bit	Name	Access	Description and value	Reset value
31	WAKEUPPADS_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
30:27	reserved	-	reserved Read value is undefined, only zero must be written.	-
26	SDMA1_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
25	reserved	-	reserved Read value is undefined, only zero must be written.	-
24	QDDKEY_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
23	CASPER_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
22	SHA_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
21	SEC_VIO_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
20	PLU_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
19	SEC_GPIO_INT01_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
18	SEC_GPIO_INT00_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0

Table 456. Start logic wake-up enable set 1 register - Offset: 6A4h)....continued

Bit	Name	Access	Description and value	Reset value
17	SEC_HYPERVISOR_CALL_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
16:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7	SPIFI_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
6	OS_EVENT_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
5	CTIMER4_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
4	CTIMER2_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
3	GPIO_INT07_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
2	GPIO_INT06_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	
1	GPIO_INT05_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0
0	GPIO_INT04_SET	R/W	Writing ones to this register sets the corresponding bit in the STARTER1 register.	0

45.4.1.2.58 Start logic wake-up enable clear 0 register (STARTERCLR0)

Table 457. Start logic wake-up enable clear 0 register - Offset: 6C0h)

Bit	Name	Access	Description and value	Reset value
31	WAKEUP_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
30	EZH_ARCH_B0_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
29	RTC_LITE0_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
28	BOD2INT_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
27	BOD1INT_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
26:24	reserved	-	reserved Read value is undefined, only zero must be written.	-
23	BLE_SLP_TMR_INT_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
22	BLE_LL_INT_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0

Table 457. Start logic wake-up enable clear 0 register - Offset: 6C0h)...continued

Bit	Name	Access	Description and value	Reset value
21:17	reserved	-	reserved Read value is undefined, only zero must be written.	-
16	FLEXINT2_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
15	FLEXINT1_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
14	FLEXINT0_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
13	CTIMER3_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
12	SCT0_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
11	CTIMER1_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
10	CTIMER0_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
9	MRT0_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
8	UTICK0_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
7	GPIO_INT03_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
6	GPIO_INT02_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
5	GPIO_INT01_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
4	GPIO_INT00_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2	GPIO_GLOBALINT0_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
1	SDMA0_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0
0	SYS_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER0 register.	0

45.4.1.2.59 Start logic wake-up enable clear 1 register (STARTERCLR1)

Table 458. Start logic wake-up enable clear 1 register - Offset: 6C4h)

Bit	Name	Access	Description and value	Reset value
31	WAKEUPPADS_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
30:27	reserved	-	reserved Read value is undefined, only zero must be written.	-
26	SDMA1_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
25	reserved	-	reserved Read value is undefined, only zero must be written.	-
24	QDDKEY_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
23	CASPER_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
22	SHA_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
21	SEC_VIO_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
20	PLU_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
19	SEC_GPIO_INT01_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
18	SEC_GPIO_INT00_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
17	SEC_HYPERVISOR_CALL_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
16:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7	SPIFI_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
6	OS_EVENT_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
5	CTIMER4_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
4	CTIMER2_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
3	GPIO_INT07_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0
2	GPIO_INT06_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0

Table 458. Start logic wake-up enable clear 1 register - Offset: 6C4h)...continued

Bit	Name	Access	Description and value	Reset value
1	GPIO_INT05_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	
0	GPIO_INT04_CLR	R/W	Writing ones to this register clears the corresponding bit in the STARTER1 register.	0

45.4.1.2.60 Functional retention control register (FUNCRETENTIONCTRL)

Table 459. Functional retention control register - Offset: 704h)

Bit	Name	Access	Description and value	Reset value
31:24	reserved	-	reserved Read value is undefined, only zero must be written.	-
23:14	RET_LENTH	R/W	length of scan chains to save during CPU retention	1
13:1	RET_START	R/W	Start address divided by 4 inside SRAMX bank	0
0	FUNCRETENA	R/W	CPU retention in power down only	0
			0b disable functional retention	
			1b enable functional retention	

45.4.1.2.61 Power-down control override signal (POWERDOWNSAFETY)

Table 460. Power-down control override signal - Offset: 70Ch)

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved Read value is undefined, only zero must be written.	-
0	OVERRIDEFRO	R/W	Overrides the fro_is_dead' signal in the sleepcon module, if it does not work on silicon.	0

45.4.1.2.62 Main clock safety cycles register (MAINCLKSAFETY)

Table 461. Main clock safety cycles register - Offset: 710h)

Bit	Name	Access	Description and value	Reset value
31:0	MAINCLKSAFETY	R/W	main clock enabled after MAINCLKSAFETY cycles	0

45.4.1.2.63 Hardware sleep control (HARDWARESLEEP)

Used to postpone power-down modes if an interrupt is pending when the processor requests deep sleep.

Table 462. Hardware sleep control - Offset: 780h)

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved Read value is undefined, only zero must be written.	-
5	SDMA1	R/W	Wake for DMA1	0
4	reserved	-	reserved Read value is undefined, only zero must be written.	-
3	SDMA0	R/W	Wake for DMA0	0
2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	PERIPHERALS	R/W	Wake for Flexcomms	0
0	FORCED	R/W	forces peripheral clocking to stay on during deep-sleep and power-down modes	0

45.4.1.2.64 CPU status (CPSTAT)

Table 463. CPU status - Offset: 80Ch

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2	CPU0LOCKUP	R/W	Arm CM33 lockup state	0
			0b CPU is not in lockup	
			1b CPU is in lockup	
1	reserved	-	reserved Read value is undefined, only zero must be written.	-
0	CPU0SLEEPING	R/W	Arm CM33 sleeping state	0
			0b CPU is not sleeping	
			1b CPU is sleeping	

45.4.1.2.65 Composite device identifier (DICE_REGn)

For n = 0 to 7.

Table 464. CPU status - Offset: 900h + (n × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	DICE_REGn	R/W	composite device identifier value that the device identifier composition engine provides after boot completion	0

45.4.1.2.66 Boot seed register (BOOT_SEED_REGn)

For n = 0 to 7.

Table 465. Boot seed register - Offset: 920h + (n × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	BOOT_SEED_REGn	R/W	256-bit random value used as boot seed	0

45.4.1.2.67 HMAC register (HMAC_REGn)

For n = 0 to 7.

Table 466. HMAC register - Offset: 940h + (n × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	HMAC_REGn	R/W	contains the calculated HMAC value	0

45.4.1.2.68 Boot seed security register (BOOT_LOCK)

Table 467. Boot seed security register - Offset: 960h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved Read value is undefined, only zero must be written.	0
1	LOCK_HMAC	R/W	control write access to HMAC_REG registers 1b write access to all 8 registers HMAC_REG is locked. This register is write once.	0
0	LOCK_BOOT_SEED	R/W	control write access to BOOT_SEED_REG registers 1b write access to all 8 registers BOOT_SEED_REG is locked. This register is write once.	0

45.4.1.2.69 Code watchdog status register (CSS_AS_CFG0)

Table 468. Code watchdog status register - Offset: 9D0h

Bit	Name	Access	Description and value	Reset value
31:14	reserved	-	reserved Read value is undefined, only zero must be written.	-
13	CFG_CWDT_ENABLED	R/W	When code watchdog timer is activated, this bit is set to 1b	0
12:0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.4.1.2.70 Clock control register (CLOCK_CTRL)

Table 469. Clock control register - Offset: A18h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved Read value is undefined, only zero must be written.	-
9	PLU_DEGLITCH_CLK_ENA	R/W	enable clocks FRO_1MHz and FRO_12MHz for PLU deglitching 0b clock is not enabled 1b clock is enabled	0
8	XO_CAL_CLK_ENA	R/W	enable clock for crystal oscillator calibration 0b clock is not enabled 1b clock is enabled	0

Table 469. Clock control register - Offset: A18h...continued

Bit	Name	Access	Description and value	Reset value
7	ANA_FRO12M_CLK_ENA	R/W	enable FRO 12 MHz clock from the FRO 192 MHz (excluding the PMC clock)	0
			0b clock is not enabled	
			1b clock is enabled	
6	FRO1MHZ_CLK_ENA	R/W	enable FRO 1 MHz clock for clock muxing in clock gen	0
			0b clock is not enabled	
			1b clock is enabled	
5	CLKIN_ENA	R/W	enable clock_in clock for clock module	0
			0b clock is not enabled	
			1b clock is enabled	
4	FRO_HF_FREQM_ENA	R/W	enable FRO 32 MHz clock for frequency measure module	0
			0b clock is not enabled	
			1b clock is enabled	
3	FRO12MHZ_FREQM_ENA	R/W	enable FRO 12 MHz clock for frequency measure module	0
			0b clock is not enabled	
			1b clock is enabled	
2	FRO1MHZ_UTICK_ENA	R/W	enable FRO 1 MHz clock for frequency measure module and for UTICK	0
			0b clock is not enabled	
			1b clock is enabled	
1	XTAL32MHZ_FREQM_ENA	R/W	enable XTAL32MHz clock for frequency measure module	0
			0b clock is not enabled	
			1b clock is enabled	
0	reserved	-	reserved Read value is undefined. Only zeros must be written.	1

45.4.1.2.71 Control automatic clock gating (AUTOCLKGATEOVERRIDE)

Table 470. Control automatic clock gating - Offset: E04h

Bit	Name	Access	Description and value	Reset value
31:16	ENABLEUPDATE	R/W	The value CODEh must be written for AUTOCLKGATEOVERRIDE registers fields updates to have effect.	0
			0000_0000_0000_0000b: bit fields 0 - 15 of this register are not updated	
			1100_0000_1101_1110b: bit fields 0 - 15 of this register are updated	
15	SYSCON	R/W	control automatic clock gating of synchronous system controller registers bank	1
			0b automatic clock gating is not overridden	
			1b automatic clock gating is overridden (clock gating is disabled)	
14	reserved	-	reserved Read value is undefined, only zero must be written.	-
13	SDMA1	R/W	control automatic clock gating of DMA1 controller	1
			0b automatic clock gating is not overridden	
			1b automatic clock gating is overridden (clock gating is disabled)	
12	SDMA0	R/W	control automatic clock gating of DMA1 controller	1
			0b automatic clock gating is not overridden	
			1b automatic clock gating is overridden (clock gating is disabled)	
11	CRCGEN	R/W	control automatic clock gating of CRCGEN controller	1
			0b automatic clock gating is not overridden	
			1b automatic clock gating is overridden (clock gating is disabled)	
10	FMC	R/W	control automatic clock gating of FMC controller	1
			0b automatic clock gating is not overridden	
			1b automatic clock gating is overridden (clock gating is disabled)	
9	FLASH	R/W	control automatic clock gating of FLASH controller	1
			0b automatic clock gating is not overridden	
			1b automatic clock gating is overridden (clock gating is disabled)	
8	SYNC1_APB	R/W	control automatic clock gating of synchronous bridge controller 1	1
			0b automatic clock gating is not overridden	
			1b automatic clock gating is overridden (clock gating is disabled)	

Table 470. Control automatic clock gating - Offset: E04h...continued

Bit	Name	Access	Description and value		Reset value
7	SYNC0_APB	R/W	control automatic clock gating of synchronous bridge controller 0		1
			0b	automatic clock gating is not overridden	
			1b	automatic clock gating is overridden (clock gating is disabled)	
6:5	reserved	-	reserved Read value is undefined, only zero must be written.		-
4	RAM2_CTRL	R/W	control automatic clock gating of RAM2 controller		1
			0b	automatic clock gating is not overridden	
			1b	automatic clock gating is overridden (clock gating is disabled)	
3	RAM1_CTRL	R/W	control automatic clock gating of RAM1 controller		1
			0b	automatic clock gating is not overridden	
			1b	automatic clock gating is overridden (clock gating is disabled)	
2	RAM0_CTRL	R/W	control automatic clock gating of RAM0 controller		1
			0b	automatic clock gating is not overridden	
			1b	automatic clock gating is overridden (clock gating is disabled)	
1	RAMX_CTRL	R/W	control automatic clock gating of RAMX controller		1
			0b	automatic clock gating is not overridden	
			1b	automatic clock gating is overridden (clock gating is disabled)	
0	ROM	R/W	control automatic clock gating of ROM controller		1
			0b	automatic clock gating is not overridden	
			1b	automatic clock gating is overridden (clock gating is disabled)	

45.4.1.2.72 GPIO synchronization control (GPIOPSYNC)

Table 471. GPIO synchronization control - Offset: E08h

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved Read value is undefined, only zero must be written.	-
0	PSYNC	R/W	enable bypass of the first stage of synchronization inside GPIO_INT module	0
			0b use the first stage of synchronization inside GPIO_INT module	
			1b bypass of the first stage of synchronization inside GPIO_INT module	

45.4.1.2.73 Invert main clock (INVERTMAINCLK)

Table 472. Invert main clock - Offset: E20h

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved Read value is undefined, only zero must be written.	-
0	INVERT	R/W	invert main_clock (AHB system clock)	0

45.4.1.2.74 HASH AES key restricted use register (HASHRESTHWKEY)

Table 473. HASH AES key restricted use register - Offset: F88h

Bit	Name	Access	Description and value	Reset value
31:0	UNLOCKCODE	R/W	Code value controlling if HASH AES hardware secret key is unlocked 1100_0011_0011_1100_1010_0101_0101_1010b: 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.	0

45.4.1.2.75 Debug lock enable (DEBUG_LOCK_EN)

Control write access to CODESECURITYPROTTEST, SWD_ACCESS_CPU0, DEBUG_FEATURES, and DEBUG_AUTH_BEACON registers. This is a write-once register.

Table 474. Debug lock enable - Offset: FA0h

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved Read value is undefined, only zero must be written.	-
3:0	LOCK_ALL	R/W	Control write access to security registers	1
			1010 enable write access to all registers	
			other disable write access to all registers	

45.4.1.2.76 Cortex debug features control (DEBUG_FEATURES)

Table 475. Cortex debug features control - Offset: FA4h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:6	CPU0_SPNIDEN	R/W	Arm CM33 secure non-invasive debug control	0
			01b any other value than 10b: invasive debug is disabled	
			10b invasive debug is enabled	
5:4	CPU0_SPIDEN	R/W	Arm CM33 secure invasive debug control	0
			01b any other value than 10b: invasive debug is disabled	
			10b invasive debug is enabled	
3:2	CPU0_NIDEN	R/W	Arm CM33 non-invasive debug control	0
			01b any other value than 10b: invasive debug is disabled	
			10b invasive debug is enabled	
1:0	CPU0_DBGGEN	R/W	Arm CM33 invasive debug control	0
			01b any other value than 10b: invasive debug is disabled	
			10b invasive debug is enabled	

45.4.1.2.77 Cortex debug features control duplicate (DEBUG_FEATURES_DP)

Table 476. Cortex debug features control duplicate - Offset: FA8h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:6	CPU0_SPNIDEN	R/W	Arm CM33 secure non-invasive debug control	0
			01b any other value than 10b: invasive debug is disabled	
			10b invasive debug is enabled	
5:4	CPU0_SPIDEN	R/W	Arm CM33 secure invasive debug control	0
			01b any other value than 10b: invasive debug is disabled	
			10b invasive debug is enabled	
3:2	CPU0_NIDEN	R/W	Arm CM33 non-invasive debug control	0
			01b any other value than 10b: invasive debug is disabled	
			10b invasive debug is enabled	
1:0	CPU0_DBGGEN	R/W	Arm CM33 invasive debug control	0
			01b any other value than 10b: invasive debug is disabled	
			10b invasive debug is enabled	

45.4.1.2.78 Security code to allow test (design for testability) access (CODESECURITYPROTTEST)

Table 477. Security code to allow test (design for testability) access - Offset: FB0h

Bit	Name	Access	Description and value	Reset value
31:0	SEC_CODE	R/W	security code to allow test access: 0x12345678	0
			0000_0000_0000_0000_0000_0000_0000_0000b: test access is not allowed	
			0001_0010_0011_0100_0101_0110_0111_1000b: security code to allow test access	

45.4.1.2.79 SWD access control register (SWD_ACCESS_CPU0)

Table 478. SWD access control register - Offset: FB4h

Bit	Name	Access	Description and value	Reset value
31:0	SEC_CODE	R/W	Arm CM33 SWD-AP: 0x12345678	0
			0000_0000_0000_0000_0000_0000_0000_0000b: CPU0 DAP is not allowed. Reading back register is read as 0x5.	
			0001_0010_0011_0100_0101_0110_0111_1000b: value to write to enable CPU0 SWD access. Reading back register will be read as 0xA.	

45.4.1.2.80 Block PUF generated keys (KEY_BLOCK)

Table 479. Block PUF generated keys - Offset: FBCh

Bit	Name	Access	Description and value	Reset value
31:0	KEY_BLOCK	R/W	Write a value to block PUF all index	0

45.4.1.2.81 Debug authentication BEACON register (DEBUG_AUTH_BEACON)

Table 480. Debug authentication BEACON register - Offset: FC0h

Bit	Name	Access	Description and value	Reset value
31:0	BEACON	R/W	Set by the debug authentication code in ROM to pass the debug beacons (Credential Beacon and authentication beacon) to application code.	0

45.4.1.2.82 Flash size configuration (FLASHSIZECFG)

Table 481. Flash size configuration - Offset: FE0h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved Read value is undefined, only zero must be written.	-
9	FLASHSTART	R/W	flash start address	0
			0b no private flash space in the main flash array	
			1b private flash start 8 kB before last address	
8:0	FLASHSIZE	R/W	flash size	1
			0_0000_000xb = 0 KB	
			0_0000_001xb = 256 KB	
			0_0000_010xb = 512 KB	
			0_0000_011xb = 768 KB	
			1_1111_1111b = 1016 KB; when others	

45.4.1.2.83 Configuration lockout register (CONFIGLOCKOUT)

Table 482. Configuration lockout register - Offset: FE4h

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved Read value is undefined, only zero must be written.	-
5	FLASHBANK3_LOCK	R/W	disable write access to the BANK3 field in the FLASHBANKENABLE register	0
			0b enable write access to the BANK3 field in the FLASHBANKENABLE register	
			1b disable write access to the BANK3 field in the FLASHBANKENABLE register	
4	FLASHBANK2_LOCK	R/W	disable write access to the BANK2 field in the FLASHBANKENABLE register	0
			0b enable write access to the BANK2 field in the FLASHBANKENABLE register	
			1b disable write access to the BANK2 field in the FLASHBANKENABLE register	
3	FLASHBANK1_LOCK	R/W	disable write access to the BANK1 field in the FLASHBANKENABLE register	0
			0b enable write access to the BANK1 field in the FLASHBANKENABLE register	
			1b disable write access to the BANK1 field in the FLASHBANKENABLE register	

Table 482. Configuration lockout register - Offset: FE4h...continued

Bit	Name	Access	Description and value	Reset value
2	FLASHBANK0_LOCK	R/W	disable write access to the BANK0 field in the FLASHBANKENABLE register	0
			0b enable write access to the BANK0 field in the FLASHBANKENABLE register	
			1b disable write access to the BANK0 field in the FLASHBANKENABLE register	
1	reserved	-	reserved Read value is undefined, only zero must be written.	-
0	LOCK	R/W	disable write access to FLASHSIZECFG, SRAMSIZECFG and PERIPHENCFG	0
			0b enable write access to FLASHSIZECFG, SRAMSIZECFG and PERIPHENCFG	
			1b disable write access to FLASHSIZECFG, SRAMSIZECFG and PERIPHENCFG	

45.4.1.2.84 RAM size configuration (RAMSIZECFG)

Table 483. RAM size configuration - Offset: FE8h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:6	RAM2_SIZE	R/W	RAM2 size	1
			00b RAM2 size = 0 KByte	
			01b RAM2 size = 8 KByte	
			10b RAM2 size = 32 KByte	
			11b RAM2 size = 64 KByte	
5:4	RAM1_SIZE	R/W	RAM1 size	1
			00b RAM1 size = 0 KByte	
			01b RAM1 size = 16 KByte	
			11b RAM1 size = 32 KByte (same for value 10b)	

Table 483. RAM size configuration - Offset: FE8h...continued

Bit	Name	Access	Description and value	Reset value
3:2	RAM0_SIZE	R/W	RAM0 size	1
			00b RAM0 size = 0 KByte	
			01b RAM0 size = 16 KByte	
			11b RAM0 size = 16 KByte (same for value 10b)	
1:0	RAMX_SIZE	R/W	RAMX size	1
			00b RAMX size = 0 KByte	
			01b RAMX size = 8 KByte	
			11b RAMX size = 16 KByte (same for value 10b)	

45.4.1.2.85 Peripheral enable configuration (PERIPHENCFG)

Table 484. Peripheral enable configuration - Offset: FECh

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15	SWFRO	R/W	Spare bit available for boot ROM to select FRO mode	1
14	PRINCEEN	R/W	enable PRINCE	1
			0b peripheral is disabled	
			1b peripheral is enabled	
13	SPIFIEN	R/W	enable SPIFI	1
			0b peripheral is disabled	
			1b peripheral is enabled	
12	HASHEN	R/W	enable HASH	1
			0b peripheral is disabled	
			1b peripheral is enabled	
11	GPIOEN	R/W	enable pins PIO0_5/6/7/8/9/10	1
			0b peripheral is disabled	
			1b peripheral is enabled	
10	reserved	-	reserved Read value is undefined, only zero must be written.	-
9	EZHBEN	R/W	enable EZH B	1
			0b peripheral is disabled	
			1b peripheral is enabled	
8	reserved	-	reserved Read value is undefined, only zero must be written.	-

Table 484. Peripheral enable configuration - Offset: FECh...continued

Bit	Name	Access	Description and value	Reset value
7	CASPEREN	R/W	enable CASPER	1
			0b peripheral is disabled	
			1b peripheral is enabled	
6	PUFFEN	R/W	enable PUF	1
			0b peripheral is disabled	
			1b peripheral is enabled	
5:4	reserved	-	reserved Read value is undefined, only zero must be written.	-
3	DMA1EN	R/W	enable DMA1	1
			0b peripheral is disabled	
			1b peripheral is enabled	
2:1	reserved	-	reserved Read value is undefined, only zero must be written.	-
0	SCTEN	R/W	enable SCT	1
			0b peripheral is disabled	
			1b peripheral is enabled	

45.4.1.2.86 Device ID (DEVICE_ID0)

Table 485. Device ID - Offset: FF8h

Bit	Name	Access	Description and value	Reset value
31:29	reserved	-	reserved Read value is undefined, only zero must be written.	-
28	DEVICE_TYPE_SEC	R	security device type Note: This bit is loaded from the SOCTRIM2.CM33_SECURITY_EXTENSION field.	0
			1 secure family	
27:24	MODELNUM_EXTENTION	R	model number extention Note: Bits 26:24 of this field is loaded from the SOCTRIM3.MODEL_NUMBER_EXTENSION field.	0
23:20	ROM_REV_MINOR	R	ROM revision	0

Table 485. Device ID - Offset: FF8h...continued

Bit	Name	Access	Description and value		Reset value
19:15	reserved	-	reserved Read value is undefined, only zero must be written.		-
14:12	FLASH_SIZE	R	flash size Note: The value of this field is loaded from the SOCTRIM2.FLASH_SIZE field.		0
			0000	flash size = 0 KB (reserved)	
			0001	flash size = 128 KB (reserved)	
			0010	flash size = 256 KB (reserved)	
			0011	flash size = 378 KB (reserved)	
			0100	flash size = 512 KB	
			0101	flash size = 640 KB (reserved)	
			0110	flash size = 768 KB (reserved)	
			0111	flash size = 896 KB (reserved)	
			1000	flash size = 1024 KB	
			other	reserved	
11:8	SRAM_SIZE	R	SRAM size Note: The value of this field is loaded from the SOCTRIM1.SRAM_SIZE field.		0
			0000	SRAM size = 128 KB	
			0001	SRAM size = 96 KB	
			0010	SRAM size = 64 KB	
			0011	SRAM size = 48 KB	
			0100	SRAM size = 32 KB	
			other	reserved	
7:0	PARTCONFIG	R	device type number		-
			0x6B	for MCX W236B	
			0x6A	for MCX W236A	
			0x5B	for MCX W235B	

45.4.1.2.87 Chip revision ID and number (DIEID)

Table 486. Device ID - Offset: FFCh

Bit	Name	Access	Description and value	Reset value
31:24	reserved	-	reserved Read value is undefined, only zero must be written.	-
23:4	MCO_NUM_IN_DIE_ID	R	chip number 0x15200	0x15200
3:0	REV_ID	R	chip metal revision ID. Note: The value of this field is loaded from the SOCTRIM1.METAL_REVISION_ID field.	4

45.4.2 I/O pin configuration (IOCON)

45.4.2.1 IOCON memory map

Table 487. IOCON - base address: 4000_1000h

Name	Access	Offset	Description	Width (bits)	Reset value	Section
PIO0_0 - PIO0_12	R/W	0h - 30h	digital I/O control for port 0 pins PIO0_b	32		Section 45.4.2.2.1
PIO0_13 - PIO0_14	R/W	34h - 38h	digital I/O control for port 0 pins PIO0_b	32		Section 45.4.2.2.2
PIO0_15 - PIO0_22	R/W	3Ch - 58h	digital I/O control for port 0 pins PIO0_b	32		Section 45.4.2.2.1

45.4.2.2 Register descriptions

45.4.2.2.1 Digital I/O control for port 0 pins PIO0_b (PIO0_0 - PIO0_12)

For b = 0 to 12 and 15 to 22

Table 488. Digital I/O control for port 0 pins PIO0_b - Offset: 0h + (b × 4h)

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved Read value is undefined, only zero must be written.	-
9	OD	R/W	controls open-drain mode	0
			0b normal; normal push-pull output	
			1b open-drain; simulated open-drain output (high drive disabled)	
8	DIGIMODE	R/W	select digital mode	0
			0b disable digital mode; digital input set to 0	
			1b enable digital mode; digital input is enabled	
7	INVERT	R/W	input polarity	0
			0b disabled; input function is not inverted	
			1b enabled; input function is inverted	
6	SLEW	R/W	driver slew rate	0
			0b standard mode; output slew rate is slower More outputs can be switched simultaneously.	
			1b fast mode; output slew rate is faster Refer to the appropriate specific device data sheet for details.	

Table 488. Digital I/O control for port 0 pins PIO0_b - Offset: 0h + (b × 4h)...continued

Bit	Name	Access	Description and value	Reset value
5:4	MODE	R/W	selects function mode (on-chip pull-up/pull-down resistor control)	0
			00b 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	
3:0	FUNC	R/W	selects pin function Note: A detailed overview on which value selects which function can be found in the MCX W23 data sheet.	0

45.4.2.2.2 Digital I/O control for port 0 pins PIO0_13 - PIO0_14)

Table 489. Offset

Register	Offset
PIO0_13	34h
PIO0_14	38h

Table 490. Digital I/O control for port 0 pins PIO0_13 and PIO0_14

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15	I2CFILTER	R/W	Configures I2C features for standard mode, fast mode, and fast mode plus operation and high-speed mode operation.	0
			0b I2C 50 ns glitch filter enabled Typically used for standard mode, fast mode, and fast mode plus I2C.	
			1b I2C 10 ns glitch filter enabled Typically used for high-speed mode I2C.	
14	EGP	R/W	switch between GPIO mode and I2C mode	1
			0b I2C mode	
			1b GPIO mode	
13	ECS	R/W	pull-up current source enable in I2C mode	0
			0b disabled; IO is in open drain cell	
			1b enabled; pull resistor is connected	

Table 490. Digital I/O control for port 0 pins PIO0_13 and PIO0_14...continued

Bit	Name	Access	Description and value		Reset value
12	FILTEROFF	R/W	controls input glitch filter		1
			0b	filter enabled Noise pulses below approximately 3 ns are filtered out in GPIO mode (EGP = 1). In I2C mode (EGP = 0), noise pulses below approximately 10 ns or 50 ns are filtered out, depending on the I2CFILTER bit field value.	
			1b	filter disabled; no input filtering is done	
11	SSEL	R/W	supply selection bit		0
			0b	3.3 V signaling in I2C mode	
			1b	1.8 V signaling in I2C Mode	
10	reserved	-	reserved Read value is undefined, only zero must be written.		-
9	OD	R/W	controls open-drain mode in standard GPIO mode (EGP = 1) This bit has no effect in I2C mode (EGP = 0).		0
			0b	normal; normal push-pull output	
			1b	open-drain; simulated open-drain output (high drive disabled)	
8	DIGIMODE	R/W	select digital mode		0
			0b	disable digital mode; digital input set to 0	
			1b	enable digital mode; digital input is enabled	
7	INVERT	R/W	input polarity		0
			0b	disabled; input function is not inverted	
			1b	enabled; input function is inverted	
6	SLEW	R/W	driver slew rate		0
			0b	standard mode; output slew rate is slower More outputs can be switched simultaneously.	
			1b	fast mode; output slew rate is faster Refer to the appropriate specific device data sheet for details.	
5:4	MODE	R/W	selects function mode (on-chip pull-up/pull-down resistor control)		0
			00b	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	
3:0	FUNC	R/W	selects pin function Note: A detailed overview on which value selects which function can be found in the MCX W23 data sheet.		0

45.4.3 Group GPIO input interrupt (GINT)

45.4.3.1 GINT memory map

Table 491. GINT - base address: 4000_1000h

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CTRL	R/W	0h	GPIO grouped interrupt control register	32		Section 45.4.3.2.1
PORT_POL	R/W	20h	GPIO grouped interrupt polarity register	32		Section 45.4.3.2.2
PORT_ENA	R/W	40h	GPIO grouped interrupt enable register	32		Section 45.4.3.2.3

45.4.3.2 Register descriptions

45.4.3.2.1 GPIO grouped interrupt control register (CTRL)

Table 492. GPIO grouped interrupt control register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2	TRIG	R/W	group interrupt trigger 0b edge-triggered 1b level-triggered	0
1	COMB	R/W	combine enabled inputs for group interrupt 0b OR functionality A grouped interrupt is generated when any the enabled inputs is active (based on its programmed polarity). 1b AND functionality An interrupt is generated when all enabled bits are active (based on their programmed polarity).	0
0	INT	R/W	group interrupt status This bit is cleared by writing a one to it. Writing zero has no effect. 0b no request; no interrupt request is pending 1b request active; interrupt request is active	0

45.4.3.2.2 GPIO grouped interrupt polarity register (PORT_POL)

Table 493. GPIO grouped interrupt polarity register - Offset: 20h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved Read value is undefined, only zero must be written.	-
22:0	POL	R/W	configure pin polarity of pins for group interrupt	1

45.4.3.2.3 GPIO grouped interrupt port 0 enable register (PORT_ENA)

Table 494. GPIO grouped interrupt port 0 enable register - Offset: 40h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved Read value is undefined, only zero must be written.	-
22:0	ENA	R/W	enable pin for group interrupt	0

45.4.4 Pin interrupt and pattern match (PINT)

45.4.4.1 PINT memory map

Table 495. PINT - base address: 4000_1000h

Name	Access	Offset	Description	Width (bits)	Reset value	Section
ISEL	R/W	0h	pin interrupt mode register	32		Section 45.4.4.2.1
IENR	R/W	4h	pin interrupt level or rising edge interrupt enable register	32		Section 45.4.4.2.2
SIENR	W	8h	pin interrupt level or rising edge interrupt set register	32		Section 45.4.4.2.3
CIENR	W	Ch	pin interrupt level or rising edge interrupt clear register	32		Section 45.4.4.2.4
IENF	R/W	10h	pin interrupt active level or falling edge interrupt enable register	32		Section 45.4.4.2.5
SIENF	W	14h	pin interrupt active level or falling edge interrupt set register	32		Section 45.4.4.2.6
CIENF	W	18h	pin interrupt active level or falling edge interrupt clear register	32		Section 45.4.4.2.7
RISE	R/W	1Ch	pin interrupt rising edge register	32		Section 45.4.4.2.8
FALL	R/W	20h	pin interrupt falling edge register	32		Section 45.4.4.2.9
IST	R/W	24h	pin interrupt status register	32		Section 45.4.4.2.10
PMCTRL	R/W	28h	pattern match interrupt control register	32		Section 45.4.4.2.11
PMSRC	R/W	2Ch	pattern match interrupt bit-slice source register	32		Section 45.4.4.2.12
PMCFG	R/W	30h	pattern match interrupt bit slice configuration register	32		Section 45.4.4.2.13

45.4.4.2 Register descriptions

45.4.4.2.1 Pin interrupt mode register (ISEL)

Table 496. Pin interrupt mode register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	PMODE	R/W	selects the interrupt mode for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn.	0
			0 edge sensitive	
			1 level sensitive	

45.4.4.2.2 Pin interrupt level or rising-edge interrupt enable register (IENR)

Table 497. Pin interrupt level or rising-edge interrupt enable register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	ENRL	R/W	enables the rising-edge interrupt or level interrupt for each pin interrupt Bit n configures the pin interrupt selected in PINTSELn.	0
			0 disable rising-edge interrupt or level interrupt	
			1 enable rising-edge interrupt or level interrupt	

45.4.4.2.3 Pin interrupt level or rising-edge interrupt set register (SIENR)

Table 498. Pin interrupt level or rising-edge interrupt set register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	SETENRL	W	ones written to this address set bits in the IENR, thus enabling interrupts Bit n sets bit n in the IENR register.	0
			0 no operation	
			1 enable rising-edge interrupt or level interrupt	

45.4.4.2.4 Pin interrupt level (rising edge interrupt) clear register (CIENR)

Table 499. Pin interrupt level (rising edge interrupt) clear register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	CENRL	W	ones written to this address clear bits in the IENR, thus disabling interrupts Bit n clears bit n in the IENR register.	0
			0 no operation	
			1 disable rising edge or level interrupt	

45.4.4.2.5 Pin interrupt active level or falling edge interrupt enable register (IENF)

Table 500. Pin interrupt active level or falling edge interrupt enable register - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	ENAF	R/W	enables the falling edge or configures the active level interrupt for each pin interrupt Bit n configures the pin interrupt selected in PINTSELn.	0
			0 disable falling-edge interrupt or set active interrupt level LOW	
			1 enable falling-edge interrupt enabled or set active interrupt level HIGH	

45.4.4.2.6 Pin interrupt active level or falling-edge interrupt clear register (CIENF)

Table 501. Pin interrupt active level or falling-edge interrupt clear register - Offset: 18h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	CENAF	W	ones written to this address clear bits in the IENF, thus enabling interrupts Bit n clears bit n in the IENF register.	0
			0 no operation	
			1 LOW-active interrupt selected or falling-edge interrupt disabled	

45.4.4.2.7 Pin interrupt active level or falling-edge interrupt clear register (CIENF)

Table 502. Pin interrupt active level or falling-edge interrupt clear register - Offset: 18h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	CENAF	W	ones written to this address clear bits in the IENF, thus disabling interrupts Bit n clears bit n in the IENF register.	0
			0 no operation	
			1 LOW-active interrupt selected or falling-edge interrupt disabled	

45.4.4.2.8 Pin interrupt rising-edge register (RISE)

Table 503. Pin interrupt rising-edge register - Offset: 1Ch

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	RDET	R/W	rising edge detect Bit n detects the rising edge of the pin selected in PINTSELn.	0
			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	

45.4.4.2.9 Pin interrupt falling-edge register (FALL)

Table 504. Pin interrupt falling-edge register - Offset: 20h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	RDET	R/W	falling edge detect Bit n detects the falling edge of the pin selected in PINTSELn.	0
			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	

45.4.4.2.10 Pin interrupt status register (IST)

Table 505. Pin interrupt status register - Offset: 24h

Bit	Name	Access	Description and value	Reset value								
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-								
7:0	PSTAT	R/W	pin interrupt status Bit n returns the status, clears the edge interrupt, or inverts the active level of the pin selected in PINTSELn. <table><tr><td>Read 0</td><td>interrupt is not requested for this interrupt pin</td></tr><tr><td>Write 0</td><td>no operation</td></tr><tr><td>Read 1</td><td>interrupt is requested for this interrupt pin</td></tr><tr><td>Write 1</td><td> edge sensitive; clear rising-edge and falling-edge detection for this pin level sensitive; switch the active level for this pin (in the IENF register) </td></tr></table>	Read 0	interrupt is not requested for this interrupt pin	Write 0	no operation	Read 1	interrupt is requested for this interrupt pin	Write 1	edge sensitive; clear rising-edge and falling-edge detection for this pin level sensitive; switch the active level for this pin (in the IENF register)	0
Read 0	interrupt is not requested for this interrupt pin											
Write 0	no operation											
Read 1	interrupt is requested for this interrupt pin											
Write 1	edge sensitive; clear rising-edge and falling-edge detection for this pin level sensitive; switch the active level for this pin (in the IENF register)											

45.4.4.2.11 Pattern match interrupt control register (PMCTRL)

Table 506. Pattern match interrupt control register - Offset: 28h

Bit	Name	Access	Description and value	Reset value	
31:24	PMAT	R/W	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.	0	
23:2	reserved	-	reserved Read value is undefined, only zero must be written.	-	
1	ENA_RXEV	R/W	enables the RXEV output to the CPU and/or to a GPIO output when the specified boolean expression evaluates to true	0	
			0b		disabled; RXEV output to the CPU is disabled
			1b		enabled; RXEV output to the CPU is enabled
0	SEL_PMATCH	R/W	specifies whether the 8 pin interrupts are controlled by the pin interrupt function or by the pattern match function	0	
			0b		pin interrupt Interrupts are driven in response to the standard pin interrupt function.
			1b		pattern match Interrupts are driven in response to pattern matches.

45.4.4.2.12 Pattern match interrupt bit-slice source register (PMSRC)

Table 507. Pattern match interrupt bit-slice source register - Offset: 2Ch

Bit	Name	Access	Description and value	Reset value
31:29	SRC7	R/W	selects the input source for bit slice 7	0
			000b input 0 Selects the pin selected in the PINTSEL0 register as the source to bit slice 7.	
			001b input 1 Selects the pin selected in the PINTSEL1 register as the source to bit slice 7.	
			010b input 2 Selects the pin selected in the PINTSEL2 register as the source to bit slice 7.	
			011b input 3 Selects the pin selected in the PINTSEL3 register as the source to bit slice 7.	
			100b input 4 Selects the pin selected in the PINTSEL4 register as the source to bit slice 7.	
			101b input 5 Selects the pin selected in the PINTSEL5 register as the source to bit slice 7.	
			110b input 6 Selects the pin selected in the PINTSEL6 register as the source to bit slice 7.	
			111b input 7 Selects the pin selected in the PINTSEL7 register as the source to bit slice 7.	
28:26	SRC6	R/W	selects the input source for bit slice 6	0
			000b input 0 Selects the pin selected in the PINTSEL0 register as the source to bit slice 6.	
			001b input 1 Selects the pin selected in the PINTSEL1 register as the source to bit slice 6.	
			010b input 2 Selects the pin selected in the PINTSEL2 register as the source to bit slice 6.	
			011b input 3 Selects the pin selected in the PINTSEL3 register as the source to bit slice 6.	

Table 507. Pattern match interrupt bit-slice source register - Offset: 2Ch...continued

Bit	Name	Access	Description and value		Reset value
28:26 (cont'd)	SRC6	R/W	100b	input 4 Selects the pin selected in the PINTSEL4 register as the source to bit slice 6.	0
			101b	input 5 Selects the pin selected in the PINTSEL5 register as the source to bit slice 6.	
			110b	input 6 Selects the pin selected in the PINTSEL6 register as the source to bit slice 6.	
			111b	input 7 Selects the pin selected in the PINTSEL7 register as the source to bit slice 6.	
25:23	SRC5	R/W	selects the input source for bit slice 5		0
			000b	input 0 Selects the pin selected in the PINTSEL0 register as the source to bit slice 5.	
			001b	input 1 Selects the pin selected in the PINTSEL1 register as the source to bit slice 5.	
			010b	input 2 Selects the pin selected in the PINTSEL2 register as the source to bit slice 5.	
			011b	input 3 Selects the pin selected in the PINTSEL3 register as the source to bit slice 5.	
			100b	input 4 Selects the pin selected in the PINTSEL4 register as the source to bit slice 5.	
			101b	input 5 Selects the pin selected in the PINTSEL5 register as the source to bit slice 5.	
			110b	input 6 Selects the pin selected in the PINTSEL6 register as the source to bit slice 5.	
			111b	input 7 Selects the pin selected in the PINTSEL7 register as the source to bit slice 5.	

Table 507. Pattern match interrupt bit-slice source register - Offset: 2Ch...continued

Bit	Name	Access	Description and value	Reset value
22:20	SRC4	R/W	selects the input source for bit slice 4	0
			000b input 0 Selects the pin selected in the PINTSEL0 register as the source to bit slice 4.	
			001b input 1 Selects the pin selected in the PINTSEL1 register as the source to bit slice 4.	
			010b input 2 Selects the pin selected in the PINTSEL2 register as the source to bit slice 4.	
			011b input 3 Selects the pin selected in the PINTSEL3 register as the source to bit slice 4.	
			100b input 4 Selects the pin selected in the PINTSEL4 register as the source to bit slice 4.	
			101b input 5 Selects the pin selected in the PINTSEL5 register as the source to bit slice 4.	
			110b input 6 Selects the pin selected in the PINTSEL6 register as the source to bit slice 4.	
			111b input 7 Selects the pin selected in the PINTSEL7 register as the source to bit slice 4.	

Table 507. Pattern match interrupt bit-slice source register - Offset: 2Ch...continued

Bit	Name	Access	Description and value	Reset value
19:17	SRC3	R/W	selects the input source for bit slice 3	0
			000b input 0 Selects the pin selected in the PINTSEL0 register as the source to bit slice 3.	
			001b input 1 Selects the pin selected in the PINTSEL1 register as the source to bit slice 3.	
			010b input 2 Selects the pin selected in the PINTSEL2 register as the source to bit slice 3.	
			011b input 3 Selects the pin selected in the PINTSEL3 register as the source to bit slice 3.	
			100b input 4 Selects the pin selected in the PINTSEL4 register as the source to bit slice 3.	
			101b input 5 Selects the pin selected in the PINTSEL5 register as the source to bit slice 3.	
			110b input 6 Selects the pin selected in the PINTSEL6 register as the source to bit slice 3.	
			111b input 7 Selects the pin selected in the PINTSEL7 register as the source to bit slice 3.	
16:14	SRC2	R/W	selects the input source for bit slice 2	0
			000b input 0 Selects the pin selected in the PINTSEL0 register as the source to bit slice 2.	
			001b input 1 Selects the pin selected in the PINTSEL1 register as the source to bit slice 2.	
			010b input 2 Selects the pin selected in the PINTSEL2 register as the source to bit slice 2.	

Table 507. Pattern match interrupt bit-slice source register - Offset: 2Ch...continued

Bit	Name	Access	Description and value		Reset value
16:14 (cont'd)	SRC2	R/W	011b	input 3 Selects the pin selected in the PINTSEL3 register as the source to bit slice 2.	0
			100b	input 4 Selects the pin selected in the PINTSEL4 register as the source to bit slice 2.	
			101b	input 5 Selects the pin selected in the PINTSEL5 register as the source to bit slice 2.	
			110b	input 6 Selects the pin selected in the PINTSEL6 register as the source to bit slice 2.	
			111b	input 7 Selects the pin selected in the PINTSEL7 register as the source to bit slice 2.	
13:11	SRC1	R/W	selects the input source for bit slice 1		0
			000b	input 0 Selects the pin selected in the PINTSEL0 register as the source to bit slice 1.	
			001b	input 1 Selects the pin selected in the PINTSEL1 register as the source to bit slice 1.	
			010b	input 2 Selects the pin selected in the PINTSEL2 register as the source to bit slice 1.	
			011b	input 3 Selects the pin selected in the PINTSEL3 register as the source to bit slice 1.	
			100b	input 4 Selects the pin selected in the PINTSEL4 register as the source to bit slice 1.	
			101b	input 5 Selects the pin selected in the PINTSEL5 register as the source to bit slice 1.	
			110b	input 6 Selects the pin selected in the PINTSEL6 register as the source to bit slice 1.	
			111b	input 7 Selects the pin selected in the PINTSEL7 register as the source to bit slice 1.	

Table 507. Pattern match interrupt bit-slice source register - Offset: 2Ch...continued

Bit	Name	Access	Description and value	Reset value
10:8	SRC0	R/W	selects the input source for bit slice 0	0
			000b input 0 Selects the pin selected in the PINTSEL0 register as the source to bit slice 0.	
			001b input 1 Selects the pin selected in the PINTSEL1 register as the source to bit slice 0.	
			010b input 2 Selects the pin selected in the PINTSEL2 register as the source to bit slice 0.	
			011b input 3 Selects the pin selected in the PINTSEL3 register as the source to bit slice 0.	
			100b input 4 Selects the pin selected in the PINTSEL4 register as the source to bit slice 0.	
			101b input 5 Selects the pin selected in the PINTSEL5 register as the source to bit slice 0.	
			110b input 6 Selects the pin selected in the PINTSEL6 register as the source to bit slice 0.	
			111b input 7 Selects the pin selected in the PINTSEL7 register as the source to bit slice 0.	
7:0	reserved	-	reserved Software must not write 1s to unused bits.	-

45.4.4.2.13 Pattern match interrupt bit slice configuration register (PMCFG)

Table 508. Pattern match interrupt bit slice configuration register - Offset: 30h

Bit	Name	Access	Description and value	Reset value
31:29	CFG7	R/W	specifies the match contribution condition for bit slice 7	0
			000b constant HIGH This bit slice always contributes to a product term match.	
			001b 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.	
			010b 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.	
			011b 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.	
			100b 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.	
			101b low level match occurs when there is a low level on the specified input.	
			110b constant 0 This bit slice never contributes to a match (should be used to disable any unused bit slices).	
			111b event Non-sticky rising or falling edge. Match occurs on an event, that is, 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 508. Pattern match interrupt bit slice configuration register - Offset: 30h...continued

Bit	Name	Access	Description and value	Reset value
28:26	CFG6	R/W	specifies the match contribution condition for bit slice 6 See bits 31:29 for bit value definition.	0
25:23	CFG5	R/W	specifies the match contribution condition for bit slice 5 See bits 31:29 for bit value definition.	0
22:20	CFG4	R/W	specifies the match contribution condition for bit slice 4 See bits 31:29 for bit value definition.	0
19:17	CFG3	R/W	specifies the match contribution condition for bit slice 3 See bits 31:29 for bit value definition.	0
16:14	CFG2	R/W	specifies the match contribution condition for bit slice 2 See bits 31:29 for bit value definition.	0
13:11	CFG1	R/W	specifies the match contribution condition for bit slice 1 See bits 31:29 for bit value definition.	0
10:8	CFG0	R/W	specifies the match contribution condition for bit slice 0 See bits 31:29 for bit value definition.	0
7	reserved	-	reserved Software must not write 1s to unused bits.	-
6	PROD_ENDPTS6	R/W	determines if slice 6 is an endpoint	0
			0b no effect Slice 6 is not an endpoint.	
			1b 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.	
5	PROD_ENDPTS5	R/W	determines if slice 5 is an endpoint	0
			0b no effect Slice 5 is not an endpoint.	
			1b 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.	
4	PROD_ENDPTS4	R/W	determines if slice 4 is an endpoint	0
			0b no effect Slice 4 is not an endpoint.	
			1b 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.	

Table 508. Pattern match interrupt bit slice configuration register - Offset: 30h...continued

Bit	Name	Access	Description and value		Reset value
3	PROD_ENDPTS3	R/W	determines if slice 3 is an endpoint		0
			0b	no effect Slice 3 is not an endpoint.	
			1b	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.	
2	PROD_ENDPTS2	R/W	determines if slice 2 is an endpoint		0
			0b	no effect Slice 2 is not an endpoint.	
			1b	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.	
1	PROD_ENDPTS1	R/W	determines if slice 1 is an endpoint		0
			0b	no effect Slice 1 is not an endpoint.	
			1b	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.	
0	PROD_ENDPTS0	R/W	determines if slice 0 is an endpoint		0
			0b	no effect Slice 0 is not an endpoint.	
			1b	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.	

45.4.5 Input multiplexing (INPUT MUX)

45.4.5.1 INPUTMUX memory map

Table 509. INPUTMUX - base address: 4000_6000h

Name	Access	Offset	Description	Width (bits)	Reset value	Section
SCT0_INMUX0 - SCT0_INMUX6	R/W	0h - 18h	input mux register for SCT0 input	32		Section 45.4.5.2.1
TIMER0CAPTSEL0 - TIMER0 CAPTSEL3	R/W	20h - 2Ch	capture select registers for TIMER0 inputs	32		Section 45.4.5.2.2
TIMER1CAPTSEL0 - TIMER1 CAPTSEL3	R/W	40h - 4Ch	capture select registers for TIMER1 inputs	32		Section 45.4.5.2.3
TIMER2CAPTSEL0 - TIMER2 CAPTSEL3	R/W	60h - 6Ch	capture select registers for TIMER2 inputs	32		Section 45.4.5.2.4
EZHARCHB_INMUX0 - EZHARCHB_INMUX7	R/W	A0h - BCh	input mux register for EZH arch B inputs	32		Section 45.4.5.2.5
PINTSEL0 - PINTSEL7	R/W	C0h - DCh	pin interrupt select register	32		Section 45.4.5.2.6
DMA0_ITRIG_INMUX0 - DMA0_ITRIG_INMUX22	R/W	E0h - 138h	trigger select register for DMA0 channel	32		Section 45.4.5.2.7
DMA0_OTRIG_INMUX0 - DMA0_OTRIG_INMUX3	R/W	160h - 16Ch	DMA0 output trigger selection to become DMA0 trigger	32		Section 45.4.5.2.8
FREQMEAS_REF	R/W	180h	selection for frequency measurement reference clock	32		Section 45.4.5.2.9
FREQMEAS_TARGET	R/W	184h	selection for frequency measurement target clock	32		Section 45.4.5.2.10
TIMER3CAPTSEL0 - TIMER3 CAPTSEL3	R/W	1A0h - 1ACh	capture select registers for TIMER3 inputs	32		Section 45.4.5.2.11
TIMER4CAPTSEL0 - TIMER4 CAPTSEL3	R/W	1C0h - 1CCh	capture select registers for TIMER4 inputs	32		Section 45.4.5.2.12
PINTSECSEL0 - PINTSECSEL1	R/W	1E0h - 1E4h	pin interrupt secure select register	32		Section 45.4.5.2.13
DMA1_ITRIG_INMUX0 - DMA1_ITRIG_INMUX9	R/W	200h - 224h	trigger select register for DMA1 channel	32		Section 45.4.5.2.14
DMA1_OTRIG_INMUX0 - DMA1_OTRIG_INMUX3	R/W	240h - 24Ch	DMA1 output trigger selection to become DMA1 trigger	32		Section 45.4.5.2.15
DMA0_REQ_ENA	R/W	740h	enable DMA0 requests	32		Section 45.4.5.2.16
DMA0_REQ_ENA_SET	W	748h	set one or several bits in DMA0_REQ_ENA register	32		Section 45.4.5.2.17
DMA0_REQ_ENA_CLR	W	750h	clear one or several bits in DMA0_REQ_ENA register	32		Section 45.4.5.2.18

Table 509. INPUTMUX - base address: 4000_6000h...continued

Name	Access	Offset	Description	Width (bits)	Reset value	Section
DMA1_REQ_ENA	R/W	760h	enable DMA1 requests	32		Section 45.4.5.2.19
DMA1_REQ_ENA_SET	W	768h	set one or several bits in DMA1_REQ_ENA register	32		Section 45.4.5.2.20
DMA1_REQ_ENA_CLR	W	770h	clear one or several bits in DMA1_REQ_ENA register	32		Section 45.4.5.2.21
DMA0_ITRIG_ENA	R/W	780h	enable DMA0 triggers	32		Section 45.4.5.2.22
DMA0_ITRIG_ENA_SET	W	788h	set one or several bits in DMA0_ITRIG_ENA register	32		Section 45.4.5.2.23
DMA0_ITRIG_ENA_CLR	W	790h	clear one or several bits in DMA0_ITRIG_ENA register	32		Section 45.4.5.2.24
DMA1_ITRIG_ENA	R/W	7A0h	enable DMA1 triggers	32		Section 45.4.5.2.25
DMA1_ITRIG_ENA_SET	W	7A8h	set one or several bits in DMA1_ITRIG_ENA register	32		Section 45.4.5.2.26
DMA1_ITRIG_ENA_CLR	W	7B0h	clear one or several bits in DMA1_ITRIG_ENA register	32		Section 45.4.5.2.27

45.4.5.2 Register descriptions

45.4.5.2.1 Input mux register for SCT0 input (SCT0_INMUX0 - SCT0_INMUX6)

Table 510. Offset

Register	Offset
SCT0_INMUX0	0h
SCT0_INMUX1	4h
SCT0_INMUX2	8h
SCT0_INMUX3	Ch
SCT0_INMUX4	10h
SCT0_INMUX5	14h
SCT0_INMUX6	18h

Table 511. Input mux register for SCT0 input - Offset: 30h

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved	-
4:0	INP_N	R/W	Input function to SCT0 inputs 0 to 6..	1
			0_0000b SCT_GPIO function selected from IOCON register	
			0_0001b SCT_GPIO1 function selected from IOCON register	
			0_0010b SCT_GPIO2 function selected from IOCON register	
			0_0011b SCT_GPIO3 function selected from IOCON register	
			0_0100b SCT_GPIO4 function selected from IOCON register	
			0_0101b SCT_GPIO5 function selected from IOCON register	
			0_0110b SCT_GPIO6 function selected from IOCON register	
			0_0111b SCT_GPIO7 function selected from IOCON register	
			0_1000b T0_OUT0 ctimer 0 match[0] output	
			0_1001b T1_OUT0 ctimer 1 match[0] output	
			0_1010b T2_OUT0 ctimer 2 match[0] output	
			0_1011b T3_OUT0 ctimer 3 match[0] output	
			0_1100b T4_OUT0 ctimer 4 match[0] output	
			0_1101b none	
			0_1110b GPIOINT_BMATCH	
			0_1111b none	
			1_0000b none	
			1_0001b none	
			1_0010b none	
			1_0011b none	

Table 511. Input mux register for SCT0 input - Offset: 30h...continued

Bit	Name	Access	Description and value		Reset value
4:0 (cont'd)	INP_N	R/W	1_0100b	none	0
			1_0101b	none	
			1_0110b	ARM_TXEV interrupt event from cpu0	
			1_0111b	DEBUG_HALTED from cpu0	
			1_1000b - 1_1111b	none	

45.4.5.2.2 Capture select registers for TIMER0 inputs (TIMER0CAPTSEL0 - TIMER0CAPTSEL3)

Table 512. Offset

Register	Offset
TIMER0CAPTSEL0	20h
TIMER0CAPTSEL1	24h
TIMER0CAPTSEL2	28h
TIMER0CAPTSEL3	2Ch

Table 513. Capture select registers for TIMER0 inputs

Bit	Name	Access	Description and value		Reset value
31:5	reserved	-	reserved		-
4:0	CAPTSEL	R/W	input function to TIMER0 capture inputs 0 to 4		1
			0_0000b	CTIMER_INP0 function selected from IOCON register	
			0_0001b	CTIMER_INP1 function selected from IOCON register	
			0_0010b	CTIMER_INP2 function selected from IOCON register	
			0_0011b	CTIMER_INP3 function selected from IOCON register	
			0_0100b	CTIMER_INP4 function selected from IOCON register	
			0_0101b	CTIMER_INP5 function selected from IOCON register	
			0_0110b	CTIMER_INP6 function selected from IOCON register	
			0_0111b	CTIMER_INP7 function selected from IOCON register	
			0_1000b	CTIMER_INP8 function selected from IOCON register	
			0_1001b	CTIMER_INP9 function selected from IOCON register	
			0_1010b	CTIMER_INP10 function selected from IOCON register	
			0_1011b	CTIMER_INP11 function selected from IOCON register	
			0_1100b	CTIMER_INP12 function selected from IOCON register	
			0_1101b	CTIMER_INP13 function selected from IOCON register	
			0_1110b	CTIMER_INP14 function selected from IOCON register	

Table 513. Capture select registers for TIMER0 inputs...continued

Bit	Name	Access	Description and value		Reset value
4:0 (cont'd)	CAPTSEL	R/W	0_1111b	CTIMER_INP15 function selected from IOCON register	1
			1_0000b	CTIMER_INP16 function selected from IOCON register	
			1_0001b	CTIMER_INP17 function selected from IOCON register	
			1_0010b	CTIMER_INP18 function selected from IOCON register	
			1_0011b	CTIMER_INP19 function selected from IOCON register	
			1_0100b	CTIMER_INP20 function selected from IOCON register	
			1_0101b	CTIMER_INP21 function selected from IOCON register	
			1_0110b	CTIMER_INP22 function selected from IOCON register	
			1_0111b - 1_1111b	none	

45.4.5.2.3 Capture select registers for TIMER1 inputs (TIMER1CAPTSEL0 - TIMER1CAPTSEL3)

Table 514. Offset

Register	Offset
TIMER1CAPTSEL0	40h
TIMER1CAPTSEL1	44h
TIMER1CAPTSEL2	48h
TIMER1CAPTSEL3	4Ch

Table 515. Capture select registers for TIMER1 inputs

Bit	Name	Access	Description and value		Reset value
31:5	reserved	-	reserved		-
4:0	CAPTSEL	R/W	input function to TIMER1 capture inputs 0 to 4		1
			0_0000b	CTIMER_INP0 function selected from IOCON register	
			0_0001b	CTIMER_INP1 function selected from IOCON register	
			0_0010b	CTIMER_INP2 function selected from IOCON register	
			0_0011b	CTIMER_INP3 function selected from IOCON register	
			0_0100b	CTIMER_INP4 function selected from IOCON register	
			0_0101b	CTIMER_INP5 function selected from IOCON register	
			0_0110b	CTIMER_INP6 function selected from IOCON register	
			0_0111b	CTIMER_INP7 function selected from IOCON register	
			0_1000b	CTIMER_INP8 function selected from IOCON register	
			0_1001b	CTIMER_INP9 function selected from IOCON register	
			0_1010b	CTIMER_INP10 function selected from IOCON register	

Table 515. Capture select registers for TIMER1 inputs...continued

Bit	Name	Access	Description and value		Reset value
4:0 (cont'd)	CAPTSEL	R/W	0_1011b	CTIMER_INP11 function selected from IOCON register	1
			0_1100b	CTIMER_INP12 function selected from IOCON register	
			0_1101b	CTIMER_INP13 function selected from IOCON register	
			0_1110b	CTIMER_INP14 function selected from IOCON register	
			0_1111b	CTIMER_INP15 function selected from IOCON register	
			1_0000b	CTIMER_INP16 function selected from IOCON register	
			1_0001b	CTIMER_INP17 function selected from IOCON register	
			1_0010b	CTIMER_INP18 function selected from IOCON register	
			1_0011b	CTIMER_INP19 function selected from IOCON register	
			1_0100b	CTIMER_INP20 function selected from IOCON register	
			1_0101b	CTIMER_INP21 function selected from IOCON register	
			1_0110b	CTIMER_INP22 function selected from IOCON register	
			1_0111b - 1_1111b	none	

45.4.5.2.4 Capture select registers for TIMER2 inputs (TIMER2CAPTSEL0 - TIMER2CAPTSEL3)

Table 516. Offset

Register	Offset
TIMER2CAPTSEL0	60h
TIMER2CAPTSEL1	64h
TIMER2CAPTSEL2	68h
TIMER2CAPTSEL3	6Ch

Table 517. Capture select registers for TIMER2 inputs

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved	-
4:0	CAPTSEL	R/W	input function to TIMER2 capture inputs 0 to 4	1
			0_0000b CTIMER_INP0 function selected from IOCON register	
			0_0001b CTIMER_INP1 function selected from IOCON register	
			0_0010b CTIMER_INP2 function selected from IOCON register	
			0_0011b CTIMER_INP3 function selected from IOCON register	
			0_0100b CTIMER_INP4 function selected from IOCON register	
			0_0101b CTIMER_INP5 function selected from IOCON register	
			0_0110b CTIMER_INP6 function selected from IOCON register	
			0_0111b CTIMER_INP7 function selected from IOCON register	
			0_1000b CTIMER_INP8 function selected from IOCON register	
			0_1001b CTIMER_INP9 function selected from IOCON register	
			0_1010b CTIMER_INP10 function selected from IOCON register	

Table 517. Capture select registers for TIMER2 inputs...continued

Bit	Name	Access	Description and value		Reset value
4:0 (cont'd)	CAPTSEL	R/W	0_1011b	CTIMER_INP11 function selected from IOCON register	1
			0_1100b	CTIMER_INP12 function selected from IOCON register	
			0_1101b	CTIMER_INP13 function selected from IOCON register	
			0_1110b	CTIMER_INP14 function selected from IOCON register	
			0_1111b	CTIMER_INP15 function selected from IOCON register	
			1_0000b	CTIMER_INP16 function selected from IOCON register	
			1_0001b	CTIMER_INP17 function selected from IOCON register	
			1_0010b	CTIMER_INP18 function selected from IOCON register	
			1_0011b	CTIMER_INP19 function selected from IOCON register	
			1_0100b	CTIMER_INP20 function selected from IOCON register	
			1_0101b	CTIMER_INP21 function selected from IOCON register	
			1_0110b	CTIMER_INP22 function selected from IOCON register	
			1_0111b - 1_1111b	none	

45.4.5.2.5 Input mux register for EZH arch B inputs (EZHARCHB_INMUXa)

For a = 0 to 7

Table 518. Input mux register for EZH arch B inputs - Offset: A0h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved	-
5:0	INP	R/W	Input number select to EZHARCHB input	1
			00_0000b - 00_1111b GPIO0_0 to GPIO0_15 functions synced to system clock, selected from IOCON registers	
			01_0000b SCT0_OUT8 output from SCTimer, synced to system clock	
			01_0001b SCT0_OUT9 output from SCTimer, synced to system clock	
			01_0010b - 01_0011b reserved	
			01_0100b MRT_IRQ0 multirate timer interrupt request 0	
			01_0101b MRT_IRQ1 multirate timer interrupt request 1	
			01_0110b T4_OUT3 ctimer 4 match[3] output	
			01_0111b T4_OUT2 ctimer 4 match[2] output	
			01_1000b T3_OUT3 ctimer 3 match[3] output	
			01_1001b T3_OUT2 ctimer 3 match[2] output	
			01_1010b T1_OUT3 ctimer 1 match[3] output	
			01_1011b T1_OUT2 ctimer 1 match[2] output	
			01_1100b UTICK0_IRQ_SYNC microtick timer interrupt, synced to system clock	
			01_1101b WDT_INT watchdog timer interrupt request, synced to system clock	
			01_1110b - 10_0101b reserved	
			10_0110b FLEXCOMM2_IRQ Flexcomm2 interrupt synced to system clock	
			10_0111b FLEXCOMM1_IRQ Flexcomm1 interrupt synced to system clock	
			10_1000b FLEXCOMM0_IRQ Flexcomm0 interrupt synced to system clock	
			10_1000b CTIMER_INP18 function selected from IOCON register	
			10_1001b DMA0_IRQ; DMA0 interrupt request	
			10_1010b DMA1_IRQ; DMA1 interrupt request	

Table 518. Input mux register for EZH arch B inputs - Offset: A0h + (a × 4h)...continued

Bit	Name	Access	Description and value		Reset value
5:0 (cont'd)	INP	R/W	10_1011b	SYS_SYNC_IRQ combined WDT_INT watchdog timer interrupt request/FLASH_IRQ flash interrupt request/BOD_IRQ brownout detection interrupt request	1
			10_1100b	RTC_IRQ real-time clock combined alarm/wake-up interrupt request	
			10_1101b	ARM_TXEV interrupt event from CPU0 or CPU1	
			10_1110b	GPIPOINT_BMATCH GPIO_INT boolean pattern match output	
			10_1111b - 11_0011b	reserved	
			11_0100b	OSTIMER_IRQ; OS timer interrupt request	
			11_0101b - 11_1111b	none	

45.4.5.2.6 Pin interrupt select register (PINTSELa)

For a = 0 to 7

Table 519. Pin interrupt select register - Offset: C0h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:7	reserved	-	reserved	-
6:0	INTPIN	R/W	pin number select for pin interrupt or pattern match engine input For PIO0_y: INTPIN = y. PIO0_0 to PIO0_22 correspond to numbers 0 to 22.	1

45.4.5.2.7 Trigger select register for DMA0 channel (DMA0_ITRIG_INMUXa)

For a = 0 to 22

Table 520. Trigger select register for DMA0 channel - Offset: E0h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:7	reserved	-	reserved	-
6:0	INP	R/W	Trigger input number (decimal value) for DMA channel n (n = 0 to 22) 0_0000b pin interrupt 0 0_0001b pin interrupt 1 0_0010b pin interrupt 2 0_0011b pin interrupt 3 0_0100b timer CTIMER0 match 0 0_0101b timer CTIMER0 match 1 0_0110b timer CTIMER1 match 0 0_0111b timer CTIMER1 match 1 0_1000b timer CTIMER2 match 0 0_1001b timer CTIMER2 match 1 0_1010b timer CTIMER3 match 0 0_1011b timer CTIMER3 match 1 0_1100b timer CTIMER4 match 0 0_1101b timer CTIMER4 match 1 0_1110b COMP_OUTPUT 0_1111b DMA0 output trigger mux 0 1_0000b DMA0 output trigger mux 1 1_0001b DMA0 output trigger mux 2 1_0010b DMA0 output trigger mux 3 1_0011b SCT0 DMA request 0 1_0100b SCT0 DMA request 1 1_0101b HASH DMA RX trigger 1_0110b - 1_1111b none	1

45.4.5.2.8 DMA0 output trigger selection to become DMA0 trigger (DMA0_OTRIG_INMUX0 - DMA0_OTRIG_INMUX3)

Table 521. Offset

Register	Offset
DMA0_OTRIG_INMUX0	160h
DMA0_OTRIG_INMUX1	164h
DMA0_OTRIG_INMUX2	168h
DMA0_OTRIG_INMUX3	16Ch

Table 522. DMA0 output trigger selection to become DMA0 trigger

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved	-
4:0	INP	R/W	DMA trigger output number (decimal value) for DMA channel n (n = 0 to 22)	1

45.4.5.2.9 Selection for frequency measurement reference clock (FREQMEAS_REF)

Table 523. Selection for frequency measurement reference clock
- Offset: 180h

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved	-
4:0	CLKIN	R/W	clock source number (decimal value) for frequency measure function target clock	1
			0_0000b 32 MHz crystal clock	
			0_0001b 12 MHz FRO clock	
			0_0010b 32 MHz FRO clock	
			0_0011b 1 MHz FRO clock	
			0_0100b 32.768 kHz RTC clock	
			0_0101b system AHB clock	
			0_0110b SPIFI clock	
			0_0111b PIO0_3 input pin	
			0_1000b PIO0_11 input pin	
			0_1001b PIO0_12 input pin	
			0_1010b PIO0_14 input pin	
			0_1101b RINGO_0 oscillator	
			0_1110b RINGO_1 oscillator	
			0_1111b RINGO_2 oscillator	

45.4.5.2.10 Selection for frequency measurement target clock (FREQMEAS_TARGET)

Table 524. Selection for frequency measurement target clock - Offset: 184h

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved	-
4:0	CLKIN	R/W	clock source number (decimal value) for frequency measure function target clock	1
			0_0000b 32 MHz crystal clock	
			0_0001b 12 MHz FRO clock	
			0_0010b 32 MHz FRO clock	
			0_0011b 1 MHz FRO clock	
			0_0100b 32.768 kHz RTC clock	
			0_0101b system AHB clock	
			0_0110b SPIFI clock	
			0_0111b PIO0_3 input pin	
			0_1000b PIO0_11 input pin	
			0_1001b PIO0_12 input pin	
			0_1010b PIO0_14 input pin	
			0_1101b RINGO_0 oscillator	
			0_1110b RINGO_1 oscillator	
			0_1111b RINGO_2 oscillator	

45.4.5.2.11 Capture select registers for TIMER3 inputs (TIMER3CAPTSEL0 - TIMER3CAPTSEL3)

Table 525. Offset

Register	Offset
TIMER3CAPTSEL0	1A0h
TIMER3CAPTSEL1	1A4h
TIMER3CAPTSEL2	1A8h
TIMER3CAPTSEL3	1ACh

Table 526. Capture select registers for TIMER3 inputs

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved	-
4:0	CAPTSEL	R/W	input function to TIMER3 capture inputs 0 to 4	1
			0_0000b CTIMER_INP0 function selected from IOCON register	
			0_0001b CTIMER_INP1 function selected from IOCON register	
			0_0010b CTIMER_INP2 function selected from IOCON register	
			0_0011b CTIMER_INP3 function selected from IOCON register	
			0_0100b CTIMER_INP4 function selected from IOCON register	
			0_0101b CTIMER_INP5 function selected from IOCON register	
			0_0110b CTIMER_INP6 function selected from IOCON register	
			0_0111b CTIMER_INP7 function selected from IOCON register	
			0_1000b CTIMER_INP8 function selected from IOCON register	
			0_1001b CTIMER_INP9 function selected from IOCON register	
			0_1010b CTIMER_INP10 function selected from IOCON register	
			0_1011b CTIMER_INP11 function selected from IOCON register	
			0_1100b CTIMER_INP12 function selected from IOCON register	
			0_1101b CTIMER_INP13 function selected from IOCON register	
			0_1110b CTIMER_INP14 function selected from IOCON register	
			0_1111b CTIMER_INP15 function selected from IOCON register	
			1_0000b CTIMER_INP16 function selected from IOCON register	
			1_0001b CTIMER_INP17 function selected from IOCON register	
			1_0010b CTIMER_INP18 function selected from IOCON register	
			1_0011b CTIMER_INP19 function selected from IOCON register	
			1_0100b CTIMER_INP20 function selected from IOCON register	
			1_0101b CTIMER_INP21 function selected from IOCON register	
			1_0110b CTIMER_INP22 function selected from IOCON register	
			1_0111b - 1_1111b none	

45.4.5.2.12 Capture select registers for TIMER4 inputs (TIMER4CAPTSEL0 - TIMER4CAPTSEL3)

Table 527. Offset

Register	Offset
TIMER4CAPTSEL0	1C0h
TIMER4CAPTSEL1	1C4h
TIMER4CAPTSEL2	1C8h
TIMER4CAPTSEL3	1CCh

Table 528. Capture select registers for TIMER4 inputs

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved	-
4:0	CAPTSEL	R/W	input function to TIMER4 capture inputs 0 to 4	1
			0_0000b CTIMER_INP0 function selected from IOCON register	
			0_0001b CTIMER_INP1 function selected from IOCON register	
			0_0010b CTIMER_INP2 function selected from IOCON register	
			0_0011b CTIMER_INP3 function selected from IOCON register	
			0_0100b CTIMER_INP4 function selected from IOCON register	
			0_0101b CTIMER_INP5 function selected from IOCON register	
			0_0110b CTIMER_INP6 function selected from IOCON register	
			0_0111b CTIMER_INP7 function selected from IOCON register	
			0_1000b CTIMER_INP8 function selected from IOCON register	
			0_1001b CTIMER_INP9 function selected from IOCON register	
			0_1010b CTIMER_INP10 function selected from IOCON register	
			0_1011b CTIMER_INP11 function selected from IOCON register	
			0_1100b CTIMER_INP12 function selected from IOCON register	
			0_1101b CTIMER_INP13 function selected from IOCON register	
			0_1110b CTIMER_INP14 function selected from IOCON register	
			0_1111b CTIMER_INP15 function selected from IOCON register	
			1_0000b CTIMER_INP16 function selected from IOCON register	
			1_0001b CTIMER_INP17 function selected from IOCON register	
			1_0010b CTIMER_INP18 function selected from IOCON register	
			1_0011b CTIMER_INP19 function selected from IOCON register	
			1_0100b CTIMER_INP20 function selected from IOCON register	
			1_0101b CTIMER_INP21 function selected from IOCON register	
			1_0110b CTIMER_INP22 function selected from IOCON register	
			1_0111b - 1_1111b none	

45.4.5.2.13 Pin interrupt secure select register (PINTSECSEL0 - PINTSECSEL1)

Table 529. Offset

Register	Offset
PINTSECSEL0	1C0h
PINTSECSEL1	1C4h

Table 530. Pin interrupt secure select register

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved	-
5:0	INTPIN	R/W	pin number select for pin interrupt secure or pattern match engine input For PIO0_x: INTPIN = x. PIO0_0 to PIO0_22 correspond to numbers 0 to 22.	1

45.4.5.2.14 Trigger select register for DMA1 channel (DMA1_ITRIG_INMUXa)

For a = 0 to 9

Table 531. Trigger select register for DMA0 channel - Offset: E0h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved	-
3:0	INP	R/W	Trigger input number (decimal value) for DMA channel n (n = 0 to 9)	1
			0_0000b pin interrupt 0	
			0_0001b pin interrupt 1	
			0_0010b pin interrupt 2	
			0_0011b pin interrupt 3	
			0_0100b timer CTIMER0 match 0	
			0_0101b timer CTIMER0 match 1	
			0_0110b timer CTIMER2 match 0	
			0_0111b timer CTIMER4 match 0	
			0_1000b DMA1 output trigger mux 0	
			0_1001b DMA1 output trigger mux 1	
			0_1010b DMA1 output trigger mux 2	
			0_1011b DMA1 output trigger mux 3	
			0_1100b SCT1 DMA request 0	
			0_1101b SCT0 DMA request 1	
			0_1110b HASH DMA RX trigger	
			0_1111b none	

45.4.5.2.15 DMA1 output trigger selection to become DMA1 trigger (DMA1_OTRIG_INMUX0 - DMA1_OTRIG_INMUX3)

Table 532. Offset

Register	Offset
DMA1_OTRIG_INMUX0	240h
DMA1_OTRIG_INMUX1	244h
DMA1_OTRIG_INMUX2	248h

Table 532. Offset...continued

Register	Offset
DMA1_OTRIG_INMUX3	24Ch

Table 533. DMA1 output trigger selection to become DMA1 trigger

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved	-
3:0	INP	R/W	DMA trigger output number (decimal value) for DMA channel n (n = 0 to 9)	1

45.4.5.2.16 Enable DMA0 requests (DMA0_REQ_ENA)

Table 534. Enable DMA0 requests - Offset: 740h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved	-
22	REQ_ENA22	R/W	spare channel; no request connected	1
21	REQ_ENA21	R/W	spare channel; no request connected	1
20	REQ_ENA20	R/W	spare channel; no request connected	1
19	REQ_ENA19	R/W	spare channel; no request connected	1
18	REQ_ENA18	R/W	spare channel; no request connected	1
17	REQ_ENA17	R/W	spare channel; no request connected	1
16	REQ_ENA16	R/W	spare channel; no request connected	1
15	REQ_ENA15	R/W	spare channel; no request connected	1
14	REQ_ENA14	R/W	spare channel; no request connected	1
13	REQ_ENA13	R/W	spare channel; no request connected	1
12	REQ_ENA12	R/W	spare channel; no request connected	1
11	REQ_ENA11	R/W	spare channel; no request connected	1
10	REQ_ENA10	R/W	spare channel; no request connected	1
9	REQ_ENA9	R/W	Flexcomm interface 2 TX/I2C master	1
8	REQ_ENA8	R/W	Flexcomm interface 2 RX/I2C slave	1
7	REQ_ENA7	R/W	Flexcomm interface 1 TX/I2C master	1
6	REQ_ENA6	R/W	Flexcomm interface 1 RX/I2C slave	1
5	REQ_ENA5	R/W	Flexcomm interface 0 TX/I2C master	1
4	REQ_ENA4	R/W	Flexcomm interface 0 RX/I2C slave	1
3	REQ_ENA3	R/W	SPIFI	1
2	REQ_ENA2	R/W	spare channel; no request connected	1
1	REQ_ENA1	R/W	spare channel; no request connected	1
0	REQ_ENA0	R/W	hash-crypt DMA request	1

45.4.5.2.17 Set one or several bits in DMA0_REQ_ENA register (DMA0_REQ_ENA_SET)

Table 535. Set one or several bits in DMA0_REQ_ENA register - Offset: 748h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved	-
22:0	SET	W	If bit #i = 1, bit #i in DMA0_REQ_ENA register is set to 1; if bit #i = 0, no change in DMA0_REQ_ENA register	0

45.4.5.2.18 Clear one or several bits in DMA0_REQ_ENA register (DMA0_REQ_ENA_CLR)

Table 536. Clear one or several bits in DMA0_REQ_ENA register - Offset: 750h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved	-
22:0	CLR	W	If bit #i = 1, bit #i in DMA0_REQ_ENA register is reset to 0; if bit #i = 0, no change in DMA0_REQ_ENA register	0

45.4.5.2.19 Enable DMA1 requests (DMA1_REQ_ENA)

Table 537. Enable DMA1 requests - Offset: 760h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved	-
9	REQ_ENA9	R/W	Flexcomm interface 2 TX/I2C master	1
8	REQ_ENA8	R/W	Flexcomm interface 2 RX/I2C master	1
7	REQ_ENA7	R/W	Flexcomm interface 1 TX/I2C master	1
6	REQ_ENA6	R/W	Flexcomm interface 1 RX/I2C master	1
5	REQ_ENA5	R/W	Flexcomm interface 0 TX/I2C master	1
4	REQ_ENA4	R/W	Flexcomm interface 0 RX/I2C master	1
3	REQ_ENA3	R/W	SPIFI	1
2	REQ_ENA2	R/W	spare channel; no request connected	1
1	REQ_ENA1	R/W	spare channel; no request connected	1
0	REQ_ENA0	R/W	hash-crypt DMA request	1

45.4.5.2.20 Set one or several bits in DMA1_REQ_ENA register (DMA1_REQ_ENA_SET)

Table 538. Set one or several bits in DMA1_REQ_ENA register - Offset: 768h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved	-
9:0	SET	W	If bit #i = 1, bit #i in DMA1_REQ_ENA register is set to 1; if bit #i = 0, no change in DMA1_REQ_ENA register	0

45.4.5.2.21 Clear one or several bits in DMA1_REQ_ENA register (DMA1_REQ_ENA_CLR)

Table 539. Clear one or several bits in DMA1_REQ_ENA register - Offset: 770h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved	-
9:0	CLR	W	If bit #i = 1, bit #i in DMA1_REQ_ENA register is reset to 0; if bit #i = 0, no change in DMA1_REQ_ENA register	0

45.4.5.2.22 Enable DMA0 triggers (DMA0_ITRIG_ENA)

Table 540. Enable DMA0 triggers - Offset: 780h

Bit	Name	Access	Description and value	Reset value
31:22	reserved	-	reserved	-
21:0	ITRIG_ENA	R/W	controls the 22 trigger inputs of DMA0. If bit i is '1' the DMA trigger input #i is enabled.	1

45.4.5.2.23 Set one or several bits in DMA0_ITRIG_ENA register (DMA0_ITRIG_ENA_SET)

Table 541. Set one or several bits in DMA0_ITRIG_ENA register - Offset: 788h

Bit	Name	Access	Description and value	Reset value
31:22	reserved	-	reserved	-
21:0	SET	W	If bit #i = 1, bit #i in DMA0_ITRIG_ENA register is set to 1; if bit #i = 0, no change in DMA0_ITRIG_ENA register	0

45.4.5.2.24 Clear one or several bits in DMA0_ITRIG_ENA register (DMA0_ITRIG_ENA_CLR)

Table 542. Clear one or several bits in DMA0_ITRIG_ENA register - Offset: 790h

Bit	Name	Access	Description and value	Reset value
31:22	reserved	-	reserved	-
21:0	CLR	W	If bit #i = 1, bit #i in DMA0_ITRIG_ENA register is reset to 0; if bit #i = 0, no change in DMA0_ITRIG_ENA register	0

45.4.5.2.25 Enable DMA1 triggers (DMA1_ITRIG_ENA)

Table 543. Enable DMA1 triggers - Offset: 7A0h

Bit	Name	Access	Description and value	Reset value
31:15	reserved	-	reserved	-
14:0	ITRIG_ENA	R/W	controls the 15 trigger inputs of DMA1. If bit i is '1' the DMA trigger input #i is enabled.	1

45.4.5.2.26 Set one or several bits in DMA1_ITRIG_ENA register (DMA1_ITRIG_ENA_SET)

Table 544. Set one or several bits in DMA1_ITRIG_ENA register - Offset: 7A8h

Bit	Name	Access	Description and value	Reset value
31:15	reserved	-	reserved	-
14:0	SET	W	If bit #i = 1, bit #i in DMA1_ITRIG_ENA register is set to 1; if bit #i = 0 , no change in DMA1_ITRIG_ENA register	0

45.4.5.2.27 Clear one or several bits in DMA1_ITRIG_ENA register (DMA1_ITRIG_ENA_CLR)

Table 545. Clear one or several bits in DMA1_ITRIG_ENA register - Offset: 7B0h

Bit	Name	Access	Description and value	Reset value
31:15	reserved	-	reserved	-
14:0	CLR	W	If bit #i = 1, bit #i in DMA1_ITRIG_ENA register is reset to 0; if bit #i = 0 , no change in DMA1_ITRIG_ENA register	0

45.4.6 Standard counter/timers (CTIMER0 to 4)

45.4.6.1 CTIMER memory map

- The CTIMER0 registers are located at base address: 4000_8000h.
- The CTIMER1 registers are located at base address: 4000_9000h.
- The CTIMER2 registers are located at base address: 4002_8000h.
- The CTIMER3 registers are located at base address: 4002_9000h.
- The CTIMER4 registers are located at base address: 4002_A000h.

Table 546. CTIMER

Name	Access	Offset	Description	Width (bits)	Reset value	Section
IR	R/W	0h	interrupt register	32		Section 45.4.6.2.1
TCR	R/W	4h	timer control register	32		Section 45.4.6.2.2
TC	R/W	8h	timer counter	32	0000_0000h	Section 45.4.6.2.3
PR	R/W	Ch	prescale register	32	0000_0000h	Section 45.4.6.2.4
PC	R/W	10h	prescale counter	32	0000_0000h	Section 45.4.6.2.5
MCR	R/W	14h	match control register	32		Section 45.4.6.2.6
MR0 - MR3	R/W	18h - 24h	match register	32	0000_0000h	Section 45.4.6.2.7
CCR	R/W	28h	capture control register	32		Section 45.4.6.2.8
CR0 - CR3	R	2Ch - 38h	capture register	32	0000_0000h	Section 45.4.6.2.9
EMR	R/W	3Ch	external match register	32		Section 45.4.6.2.10
CTCR	R/W	70h	count control register	32		Section 45.4.6.2.11
PWMC	R/W	74h	PWM Control Register	32		Section 45.4.6.2.12
MSR0 - MSR3	r/w	78h - 84h	match shadow register	32	0000_0000h	Section 45.4.6.2.13

45.4.6.2 Register descriptions

45.4.6.2.1 Interrupt register (IR)

The interrupt register consists of 4 bits for the match interrupts and 4 bits for the capture interrupts. If an interrupt is generated, the corresponding bit in the IR is high. Otherwise, the bit is low. Writing a logic one to the corresponding IR bit resets 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 547. Interrupt register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved	-
7	CR3INT	R/W	interrupt flag for capture channel 3 event	0
6	CR2INT	R/W	interrupt flag for capture channel 2 event	0
5	CR1INT	R/W	interrupt flag for capture channel 1 event	0
4	CR0INT	R/W	interrupt flag for capture channel 0 event	0
3	MR3INT	R/W	interrupt flag for match channel 3	0
2	MR2INT	R/W	interrupt flag for match channel 2	0
1	MR1INT	R/W	interrupt flag for match channel 1	0
0	MR0INT	R/W	interrupt flag for match channel 0	0

45.4.6.2.2 Timer control register (TCR)

The TCR is used to control the timer counter functions. The timer counter can be disabled or reset through the TCR.

Table 548. Timer control register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1	CRST	R/W	counter reset	0
			0b disabled Do nothing.	
			1b 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.	
0	CEN	R/W	counter enable	0
			0b disabled The counters are disabled.	
			1b enabled The timer counter and prescale counter are enabled.	

45.4.6.2.3 Timer counter (TC)

When the prescale counter reaches its terminal count, the 32-bit timer counter register is incremented. Unless it is reset before reaching its upper limit, the timer counter counts up through the value 0xFFFF FFFF and then wraps back to the value 0x0000 0000. This event does not cause an interrupt. However, a match register can be used to detect an overflow, if necessary.

Table 549. Timer counter - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:0	TCVAL	R/W	timer counter value	0

45.4.6.2.4 Prescale register (PR)

The 32-bit prescale register specifies the maximum value for the prescale counter.

Table 550. Prescale register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:0	PRVAL	R/W	prescale counter value	0

45.4.6.2.5 Prescale counter (PC)

The 32-bit prescale counter controls division of the APB bus clock by some constant value before it is applied to the timer counter. It allows the control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The prescale counter is incremented on every APB bus clock. When it reaches the value stored in the prescale register, the timer counter is incremented and the prescale counter is reset on the next APB bus clock. It causes the timer counter to increment on every APB bus clock when PR = 0, every 2 APB bus clocks when PR = 1, and so on.

Table 551. Prescale counter - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:0	PCVAL	R/W	prescale counter value	-

45.4.6.2.6 Match control register (MCR)

The match control register is used to control what operations are performed when one of the match registers matches the timer counter.

Table 552. Match control register - Offset: 14h

Bit	Name	Access	Description and value	Reset value
31:28	reserved	-	reserved Read value is undefined, only zero must be written.	-
27	MR3RL	R/W	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
26	MR2RL	R/W	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

Table 552. Match control register - Offset: 14h...continued

Bit	Name	Access	Description and value	Reset value
25	MR1RL	R/W	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
24	MR0RL	R/W	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
23:12	reserved	-	reserved Read value is undefined, only zero must be written.	-
11	MR3S	R/W	stop on MR3 The TC and PC is stopped and TCR[0] is set to 0 if MR3 matches the TC.	0
10	MR3R	R/W	reset on MR3 The TC is reset if MR3 matches it.	0
9	MR3I	R/W	interrupt on MR3 An interrupt is generated when MR3 matches the value in the TC.	0
8	MR2S	R/W	stop on MR2 The TC and PC is stopped and TCR[0] is set to 0 if MR2 matches the TC.	0
7	MR2R	R/W	reset on MR2 The TC is reset if MR2 matches it.	0
6	MR2I	R/W	interrupt on MR2 An interrupt is generated when MR2 matches the value in the TC.	0
5	MR1S	R/W	stop on MR1 The TC and PC is stopped and TCR[0] is set to 0 if MR1 matches the TC.	0
4	MR1R	R/W	reset on MR1 The TC is reset if MR1 matches it.	0
3	MR1I	R/W	interrupt on MR1 An interrupt is generated when MR1 matches the value in the TC.	0
2	MR0S	R/W	stop on MR0 The TC and PC is stopped and TCR[0] is set to 0 if MR0 matches the TC.	0
1	MR0R	R/W	reset on MR0 The TC is reset if MR0 matches it.	0
0	MR0I	R/W	interrupt on MR0 An interrupt is generated when MR0 matches the value in the TC.	0

45.4.6.2.7 Match register (MR0 - MR3)

Table 553. Offset

Register	Offset
MR0	18h
MR1	1Ch
MR2	20h
MR3	24h

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. The settings in the MCR register control the actions. If the associated MRxRL bit in the match control register is set, the match register is automatically reloaded with the current contents of its corresponding match shadow register when the TC is cleared to zero. This transfer takes 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 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 554. Match register

Bit	Name	Access	Description and value	Reset value
31:0	MATCH	R/W	Timer counter match value	0

45.4.6.2.8 Capture control register (CCR)

The capture control register is used to control if one of the four capture registers is loaded with the value in the timer counter when the capture event occurs. It is also used to control if the capture event generates an interrupt. Setting the rising bits and falling bits at the same time is a valid configuration. The result is 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 must be programmed as 000. However, capture and/or interrupt can be selected for the other three CAP inputs.

Table 555. Capture control register - Offset: 28h

Bit	Name	Access	Description and value	Reset value
31:12	reserved	-	reserved Read value is undefined, only zero must be written.	-
11	CAP3I	R/W	generate interrupt on channel 3 capture event A CR3 load generates an interrupt.	0
10	CAP3FE	R/W	falling-edge of capture channel 3 A sequence of 1 then 0 causes CR3 to be loaded with the contents of TC.	0
			0 disabled	
			1 enabled	

Table 555. Capture control register - Offset: 28h...continued

Bit	Name	Access	Description and value	Reset value
9	CAP3RE	R/W	rising-edge of capture channel 3 A sequence of 0 then 1 causes CR3 to be loaded with the contents of TC.	0
			0 disabled	
			1 enabled	
8	CAP2I	R/W	generate interrupt on channel 2 capture event A CR2 load generates an interrupt.	0
7	CAP2FE	R/W	falling-edge of capture channel 2 A sequence of 1 then 0 causes CR2 to be loaded with the contents of TC.	0
			0 disabled	
			1 enabled	
6	CAP2RE	R/W	rising-edge of capture channel 2 A sequence of 0 then 1 causes CR2 to be loaded with the contents of TC.	0
			0 disabled	
			1 enabled	
5	CAP1I	R/W	generate interrupt on channel 1 capture event A CR1 load generates an interrupt.	0
4	CAP1FE	R/W	falling-edge of capture channel 1 A sequence of 1 then 0 causes CR1 to be loaded with the contents of TC.	0
			0 disabled	
			1 enabled	
3	CAP1RE	R/W	rising-edge of capture channel 1 A sequence of 0 then 1 causes CR1 to be loaded with the contents of TC.	0
			0 disabled	
			1 enabled	
2	CAP0I	R/W	generate interrupt on channel 0 capture event A CR0 load generates an interrupt.	0
1	CAP0FE	R/W	falling-edge of capture channel 0 A sequence of 1 then 0 causes CR0 to be loaded with the contents of TC.	0
			0 disabled	
			1 enabled	
0	CAP0RE	R/W	rising-edge of capture channel 0 A sequence of 0 then 1 causes CR0 to be loaded with the contents of TC.	0
			0 disabled	
			1 enabled	

45.4.6.2.9 Capture register (CR0 - CR3)

The capture control register is used to control if one of the four capture registers is loaded with the value in the timer counter when the capture event occurs. It is also used to control if the capture event generates an interrupt. Setting the rising bits and the falling bits at the same time is a valid configuration. The result is 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 must be programmed as 000. However, capture and/or interrupt can be selected for the other three CAP inputs.

Table 556. Offset

Register	Offset
CR0	2Ch
CR1	30h
CR2	34h
CR3	38h

Table 557. Capture register

Bit	Name	Access	Description and value	Reset value
31:0	CAP	R/W	timer counter capture value	0

45.4.6.2.10 External match register (EMR)

The EMR controls the match function and the external match pins.

Table 558. External match register - Offset: 3Ch

Bit	Name	Access	Description and value	Reset value
31:12	reserved	-	reserved Read value is undefined, only zero must be written.	-
11:10	EMC3	R/W	external match control 3 Determines the functionality of external match 3.	0
			00b do nothing	
			01b clear Clear the corresponding external match bit/output to 0 (MAT3 pin is LOW if pinned out).	
			10b set Set the corresponding external match bit/output to 1 (MAT3 pin is HIGH if pinned out).	
			11b toggle Toggle the corresponding external match bit/output.	

Table 558. External match register - Offset: 3Ch...continued

Bit	Name	Access	Description and value	Reset value
9:8	EMC2	R/W	external match control 2 Determines the functionality of external match 2.	0
			00b do nothing	
			01b clear Clear the corresponding external match bit/output to 0 (MAT2 pin is LOW if pinned out).	
			10b set Set the corresponding external match bit/output to 1 (MAT2 pin is HIGH if pinned out).	
			11b toggle Toggle the corresponding external match bit/output.	
7:6	EMC1	R/W	external match control 1 Determines the functionality of external match 1.	0
			00b do nothing	
			01b clear Clear the corresponding external match bit/output to 0 (MAT1 pin is LOW if pinned out).	
			10b set Set the corresponding external match bit/output to 1 (MAT1 pin is HIGH if pinned out).	
			11b toggle Toggle the corresponding external match bit/output.	
5:4	EMC0	R/W	external match control 0 Determines the functionality of external match 0.	0
			00b do nothing	
			01b clear Clear the corresponding external match bit/output to 0 (MAT1 pin is LOW if pinned out).	
			10b set Set the corresponding external match bit/output to 1 (MAT1 pin is HIGH if pinned out).	
			11b toggle Toggle the corresponding external match bit/output.	

Table 558. External match register - Offset: 3Ch...continued

Bit	Name	Access	Description and value	Reset value
3:2	EM3	R/W	external match 3 This bit reflects the state of output MAT3, if 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
			0 LOW	
			1 HIGH	
	EM2	R/W	external match 2 This bit reflects the state of output MAT2, if 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 MR[9:8]. This bit is driven to the MAT pins if the match function is selected via IOCON.	0
			0 LOW	
			1 HIGH	
1	EM1	R/W	external match 1 This bit reflects the state of output MAT1, if 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 MR[7:6]. This bit is driven to the MAT pins if the match function is selected via IOCON.	0
			0 LOW	
			1 HIGH	
0	EM0	R/W	external match 0 This bit reflects the state of output MAT0, if 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 MR[5:4]. This bit is driven to the MAT pins if the match function is selected via IOCON.	0
			0 LOW	
			1 HIGH	

45.4.6.2.11 Count control register (CTCR)

The CTCR selects between timer and counter mode. In counter mode, it selects the signal and edges for counting.

Table 559. Count control register - Offset: 70h

Bit	Name	Access	Description and value		Reset value
31:8	reserved	-	reserved		-
7:5	SELCC	R/W	edge select When bit 4 is 1, these bits select which capture input edge causes the timer and prescaler to be cleared. These bits have no effect when bit 4 is low. Values 0x2 to 0x3 and 0x6 to 0x7 are reserved.		0
			000b	channel 0 rising edge Rising edge of the signal on capture channel 0 clears the timer (if bit 4 is set).	
			001b	channel 0 falling edge Falling edge of the signal on capture channel 0 clears the timer (if bit 4 is set).	
			010b	channel 1 rising edge Rising edge of the signal on capture channel 1 clears the timer (if bit 4 is set).	
			011b	channel 1 falling edge Falling edge of the signal on capture channel 1 clears the timer (if bit 4 is set).	
			100b	channel 2 rising edge Rising edge of the signal on capture channel 2 clears the timer (if bit 4 is set).	
			101b	channel 2 falling edge Falling edge of the signal on capture channel 2 clears the timer (if bit 4 is set).	
4	ENCC	R/W	Setting this bit to 1 enables clearing of the timer and the prescaler when the capture-edge event specified in bits 7:5 occurs.		0
3:2	CINSEL	R/W	count input select When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking. Note: <i>If counter mode is selected for a particular CAPn input in the CTCR, the 3 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 3 CAPn inputs in the same timer.</i>		0
			00b	channel 0; CAPn.0 for CTIMERn	
			01b	channel 1; CAPn.1 for CTIMERn	
			10b	channel 2; CAPn.2 for CTIMERn	
			11b	channel 3; CAPn.3 for CTIMERn	

Table 559. Count control register - Offset: 70h...continued

Bit	Name	Access	Description and value	Reset value								
1:0	CTMODE	R/W	counter/timer mode This field selects which rising APB bus clock edges can increment the prescale counter (PC) of the timer or clear PC and increment timer counter (TC). Timer mode: When the prescale counter matches the prescale register, the TC is incremented. <table><tr><td>00b</td><td>timer mode Incremented every rising APB bus clock edge.</td></tr><tr><td>01b</td><td>counter mode rising edge TC is incremented on rising edges on the CAP input selected by bits 3:2.</td></tr><tr><td>10b</td><td>counter mode falling edge TC is incremented on falling edges on the CAP input selected by bits 3:2.</td></tr><tr><td>11b</td><td>counter mode dual edge TC is incremented on both edges on the CAP input selected by bits 3:2.</td></tr></table>	00b	timer mode Incremented every rising APB bus clock edge.	01b	counter mode rising edge TC is incremented on rising edges on the CAP input selected by bits 3:2.	10b	counter mode falling edge TC is incremented on falling edges on the CAP input selected by bits 3:2.	11b	counter mode dual edge TC is incremented on both edges on the CAP input selected by bits 3:2.	0
00b	timer mode Incremented every rising APB bus clock edge.											
01b	counter mode rising edge TC is incremented on rising edges on the CAP input selected by bits 3:2.											
10b	counter mode falling edge TC is incremented on falling edges on the CAP input selected by bits 3:2.											
11b	counter mode dual edge TC is incremented on both edges on the CAP input selected by bits 3:2.											

45.4.6.2.12 PWM control register (PWMC)

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 external match register (EMR) controls. 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 match register that is configured to set the PWM cycle length resets the timer. When the timer is reset to zero, all currently HIGH match outputs configured as PWM outputs are cleared.

Table 560. PWM control register - Offset: 74h

Bit	Name	Access	Description and value		Reset value
31:4	reserved	-	reserved		-
3	PWMEN3	R/W	PWM mode enable for channel3 Note: It is recommended to use match channel 3 to set the PWM cycle.		0
			0b	Match CTIMERn_MAT3 is controlled by EM3.	
			1b	PWM PWM mode is enabled for CT132Bn_MAT3.	
2	PWMEN2	R/W	PWM mode enable for channel2 Note: It is recommended to use match channel 2 to set the PWM cycle.		0
			0b	Match CTIMERn_MAT2 is controlled by EM2.	
			1b	PWM PWM mode is enabled for CT132Bn_MAT2.	

Table 560. PWM control register - Offset: 74h...continued

Bit	Name	Access	Description and value	Reset value
1	PWMEN1	R/W	PWM mode enable for channel1 Note: It is recommended to use match channel 1 to set the PWM cycle.	0
			0b Match CTIMERn_MAT1 is controlled by EM1.	
			1b PWM PWM mode is enabled for CT132Bn_MAT1.	
0	PWMEN0	R/W	PWM mode enable for channel0 Note: It is recommended to use match channel 0 to set the PWM cycle.	0
			0b Match CTIMERn_MAT0 is controlled by EM0.	
			1b PWM PWM mode is enabled for CT132Bn_MAT0.	

45.4.6.2.13 Match shadow register (MSR0 - MSR3)

Table 561. Offset

Register	Offset
MSR0	78h
MSR1	7Ch
MSR2	80h
MSR3	84h

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) is also programmed to generate an interrupt or DMA request. Before the next reload occurs, software or the DMA engine has one full counter cycle to modify the contents of the match shadow registers.

Table 562. Match shadow register

Bit	Name	Access	Description and value	Reset value
31:0	SHADOW	R/W	timer counter match shadow value	0

45.4.7 Windowed watchdog timer (WWDT)

45.4.7.1 WWDT memory map

The WWDT registers are located at base address: 4000_C000h.

Table 563. Windowed watchdog timer registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
MOD	R/W	0h	watchdog mode register	32		Section 45.4.7.2.1
TC	R/W	4h	watchdog timer constant register	32		Section 45.4.7.2.2
FEED	W	8h	watchdog feed sequence register	32		Section 45.4.7.2.3
TV	R	Ch	watchdog timer value register	32		Section 45.4.7.2.4
WARNINT	R/W	14h	watchdog warning interrupt compare value	32		Section 45.4.7.2.5
WINDOW	R/W	18h	watchdog window compare value	32		Section 45.4.7.2.6

45.4.7.2 Register descriptions

45.4.7.2.1 Watchdog mode register (MOD)

This register contains the basic mode and status of the watchdog timer.

Table 564. Watchdog mode register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved Read value is undefined, only zero must be written.	-
4	WDPROTECT	R/W	watchdog update mode Software can set this bit once. It can only be cleared using a reset. 0b flexible When the watchdog timer is below the value of watchdog WINDOW field, the watchdog reload/watchdog feed sequence can be performed. 1b threshold The watchdog reload/watchdog feed sequence can only be performed after the watchdog timer is below the value of watchdog WARNING and watchdog WINDOW.	0
3	WDINT	R/W	warning interrupt flag Set when the timer is at or below the value in watchdog WARNINT. Cleared with software writing a 1 to this bit position. Note: <i>This bit cannot be cleared while the WARNINT value equals the value of the TV register. This event can occur if the value of WARNINT is 0 and the WDRESET bit is 0 when TV decrements to 0.</i>	0
2	WDTOF	R/W	watchdog timeout flag Set when the watchdog timer times out, because of a feed error, or because of events associated with WDPROTECT. Cleared with software writing a 0 to this bit position. If WDRESET = 1, it causes a chip reset.	0
1	WDRESET	R/W	watchdog reset enable bit When this bit has been written with a 1, it cannot be rewritten with a 0. 0b interrupt A watchdog timeout does not cause a chip reset. 1b reset A watchdog timeout causes a chip reset.	0
0	WDEN	R/W	watchdog enable bit When this bit is set to 1 and a watchdog feed is performed, the watchdog timer runs permanently. 0b stop The watchdog timer is stopped. 1b run The watchdog timer is running.	0

45.4.7.2.2 Watchdog timer constant register (TC)

This 24-bit register determines the timeout value.

Table 565. Watchdog timer constant register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:24	reserved	-	reserved Read value is undefined, only zero must be written.	-
23:0	COUNT	R/W	watchdog timeout value	0000FFh

45.4.7.2.3 Watchdog feed sequence register (FEED)

Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in TC.

Table 566. Watchdog feed sequence register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	FEED	W	Feed value must be 0xAA followed by 0x55.	-

45.4.7.2.4 Watchdog timer value register (TV)

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 the CPU reads it.

Table 567. Watchdog timer value register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:24	reserved	-	reserved Read value is undefined, only zero must be written.	-
23:0	COUNT	R	counter timer value	0000FFh

45.4.7.2.5 Watchdog warning interrupt compare value (WARNINT)

The watchdog WARNINT register determines the watchdog timer counter value that generates a watchdog interrupt. When the watchdog timer counter is no longer greater than the value that WARNINT defines, an interrupt is generated after the subsequent WDCLK. 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, a match of the watchdog timer counter to WARNINT occurs. It gives a maximum time of 1,023 watchdog timer counts (4,096 watchdog clocks) for the interrupt to occur before a watchdog event. If WARNINT is 0, the interrupt occurs at the same time as the watchdog event.

Table 568. Watchdog warning interrupt compare value - Offset: 14h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved Read value is undefined, only zero must be written.	-

Table 568. Watchdog warning interrupt compare value - Offset: 14h...continued

Bit	Name	Access	Description and value	Reset value
9:0	WARNINT	R/W	watchdog warning interrupt compare value	0

45.4.7.2.6 Watchdog window compare value (WINDOW)

When a watchdog feed is performed, the WINDOW register determines the highest TV value allowed. If a feed sequence occurs when TV is greater than the value in WINDOW, a watchdog event occurs. WINDOW resets to the maximum TV value.

Table 569. Watchdog window compare value - Offset: 18h

Bit	Name	Access	Description and value	Reset value
31:24	reserved	-	reserved Read value is undefined, only zero must be written.	-
23:0	WINDOW	R/W	watchdog window value	0

45.4.8 Multi-rate timer (MRT)

45.4.8.1 MRT memory map

The MRT registers are located at base address: 4000_D000h.

Table 570. Multirate timer registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
MRT timer 0 registers						
INTVAL0	R/W	0h	MRT time interval value register	32		Section 45.4.8.2.1
TIMER0	R	4h	MRT timer register	32		Section 45.4.8.2.2
CTRL0	R/W	8h	MRT control register	32		Section 45.4.8.2.3
STAT0	R/W	Ch	MRT status register	32		Section 45.4.8.2.4
MRT timer 1 registers						
INTVAL1	R/W	10h	MRT time interval value register	32		Section 45.4.8.2.1
TIMER1	R	14h	MRT timer register	32		Section 45.4.8.2.2
CTRL1	R/W	18h	MRT control register	32		Section 45.4.8.2.3
STAT1	R/W	1Ch	MRT status register			Section 45.4.8.2.4
MRT timer 2 registers						
INTVAL2	R/W	20h	MRT time interval value register	32		Section 45.4.8.2.1
TIMER2	R	24h	MRT timer register	32		Section 45.4.8.2.2
CTRL2	R/W	28h	MRT control register	32		Section 45.4.8.2.3
STAT2	R/W	2Ch	MRT status register	32		Section 45.4.8.2.4
MRT timer 3 registers						
INTVAL3	R/W	30h	MRT time interval value register	32		Section 45.4.8.2.1
TIMER3	R	34h	MRT timer register	32		Section 45.4.8.2.2
CTRL3	R/W	38h	MRT control register	32		Section 45.4.8.2.3
STAT3	R/W	3Ch	MRT status register			Section 45.4.8.2.4
Global MRT registers						
MODCFG	R/W	F0h	module configuration register	32		Section 45.4.8.2.5
IDLE_CH	R	F4h	idle channel register	32		Section 45.4.8.2.6
IRQ_FLAG	R/W	F8h	global interrupt flag register			Section 45.4.8.2.7

45.4.8.2 Register descriptions

45.4.8.2.1 MRT time interval value register (INTVAL0 - INTVAL3)

This register contains the MRT load value and controls how the timer is reloaded. The load value is IVALUE – 1.

Table 571. Offset

Register	Offset
INTVAL0	0h
INTVAL1	10h
INTVAL2	20h
INTVAL3	30h

Table 572. MRT time interval value register

Bit	Name	Access	Description and value	Reset value
31	LOAD	W	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 If the repeat mode is selected, the load from the INTVALn register to the TIMERN register is processed at the end of the time interval.	
			1 force load While TIMERN is in active state, the INTVALn interval value IVALUE – 1 is immediately loaded into the TIMERN register.	
30:24	reserved	-	reserved Read value is undefined, only zero must be written.	-
23:0	IVALUE	R/W	time interval load value This value is loaded into the TIMERN register. 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

45.4.8.2.2 MRT timer register (TIMER0 - TIMER3)

The timer register holds the current timer value. This register is read-only.

Table 573. Offset

Register	Offset
TIMER0	4h
TIMER1	14h
TIMER2	24h
TIMER3	34h

Table 574. MRT timer register

Bit	Name	Access	Description and value	Reset value
31:24	reserved	-	reserved Read value is undefined, only zero must be written.	-
23:0	VALUE	R	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).	0

45.4.8.2.3 MRT control register (CTRL0 - CTRL3)

The control register configures the mode for each MRT and enables the interrupt.

Table 575. Offset

Register	Offset
CTRL0	8h
CTRL1	18h
CTRL2	28h
CTRL3	38h

Table 576. MRT control register

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2:1	MODE	R/W	selects timer mode	0
			00b repeat interrupt mode	
			01b one-shot interrupt mode	
			10b one-shot stall mode	
			11b reserved	
0	INTEN	R/W	enable the TIMERN interrupt	0
			0b disabled TIMERN interrupt is disabled.	
			1b enabled TIMERN interrupt is enabled.	

45.4.8.2.4 MRT status register (STAT0 - STAT3)

This register indicates the status of each MRT.

Table 577. Offset

Register	Offset
STAT0	Ch
STAT1	1Ch
STAT2	2Ch
STAT3	3Ch

Table 578. MRT control register

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2	INUSE	R/W	channel in use flag Operating details depend on the MULTITASK bit in the MODCFG register. They affect the use of IDLE_CH. For details of the two operating modes, see Section 45.4.8.2.5 .	0
			0b this channel is not in use	
			1b this channel is in use	

Table 578. MRT control register...continued

Bit	Name	Access	Description and value	Reset value
1	RUN	R	indicates the state of TIMERN This bit is read-only.	0
			0b idle state TIMERN is stopped.	
			1b running TIMERN is running.	
0	INTFLAG	R/W	monitors the interrupt flag	0
			0b no pending interrupt writing a zero is equivalent to no operation.	
			1b 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.	

45.4.8.2.5 Module configuration register (MODCFG)

The MODCFG register provides the configuration (number of channels and timer width) for this MRT.

Table 579. Module configuration register - Offset: F0h

Bit	Name	Access	Description and value	Reset value
31	MULTITASK	R/W	selects the operating mode for the INUSE flags and the IDLE_CH register	0
			0b hardware status mode In this mode, the INUSE(n) flags for all channels are reset.	
			1b multitask mode	
30:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8:4	NOB	R	identifies the number of timer bits in this MRT (24 bits wide on this device)	17h
3:0	NOC	R	identifies the number of channels in this MRT (4 channels on this device)	3h

45.4.8.2.6 Idle channel register (IDLE_CH)

This register returns the number of the first idle channel.

Table 580. Idle channel register - Offset: F4h

Bit	Name	Access	Description and value	Reset value
30:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:4	CHAN	R	idle channel Reading the CHAN bits, returns the lowest idle timer channel. The number is positioned such that, to access the registers for the allocated channel, it can be used as an offset from the MRT base address . If all timer channels are running, CHAN = 0xF.	0
3:0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.4.8.2.7 Global interrupt flag register (IRQ_FLAG)

Table 581. Global interrupt flag register - Offset: F8h

Bit	Name	Access	Description and value	Reset value
30:4	reserved	-	reserved Read value is undefined, only zero must be written.	-
3	GFLAG3	R/W	monitors the interrupt flag of TIMER3 See description of channel 0.	0
2	GFLAG2	R/W	monitors the interrupt flag of TIMER2 See description of channel 0.	0
1	GFLAG1	R/W	monitors the interrupt flag of TIMER1 See description of channel 0.	0
0	GFLAG0	R/W	monitors the interrupt flag of TIMER0	0
			0b no pending interrupt Writing a zero is equivalent to no operation.	
			1b pending interrupt The interrupt is pending because TIMER0 has reached the end of the time interval. If the INTEN bit in the CTRL0 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.	

45.4.9 Micro-tick timer (UTICK)

45.4.9.1 UTICK memory map

The UTICK registers are located at base address: 4000_E000h.

Table 582. Micro-tick timer registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CTRL	R/W	0h	control register	32	0000_0000h	Section 45.4.9.2.1
STAT	R/W	4h	status register	32		Section 45.4.9.2.2
CFG	R/W	8h	capture configuration register	32		Section 45.4.9.2.3
CAPCLR	W	Ch	capture clear register	32		Section 45.4.9.2.4
CAP0 - CAP3	R	10h - 1Ch	capture register	32	0000_0000h	Section 45.4.9.2.5

45.4.9.2 Register descriptions

45.4.9.2.1 Control register (CTRL)

This register controls the micro-tick timer. Any write to the CTRL register resets the counter, meaning a new interval is measured if one was in progress.

Table 583. Control register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31	REPEAT	R/W	repeat delay	0
			0 one-time delay	
			1 delay repeats continuously	
30:0	DELAYVAL	R/W	tick interval value The delay equals 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

45.4.9.2.2 Status register (STAT)

This register provides status for the micro-tick timer.

Table 584. Status register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1	ACTIVE	R/W	active flag	0
			0 The micro-tick timer is stopped.	
			1 The micro-tick timer is currently running.	
0	INTR	R/W	interrupt flag A write of any value to this register clears this flag.	0
			0 no interrupt is pending	
			1 interrupt is pending	

45.4.9.2.3 Capture configuration register (CFG)

This register allows enabling micro-tick capture functions and selects the polarity of the capture triggers.

Table 585. Capture configuration register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:12	reserved	-	reserved	-
11	CAPPOL3	R/W	capture polarity 3	0
			0 positive edge capture	
			1 negative edge capture	
10	CAPPOL2	R/W	capture polarity 2	0
			0 positive edge capture	
			1 negative edge capture	
9	CAPPOL1	R/W	capture polarity 1	0
			0 positive edge capture	
			1 negative edge capture	
8	CAPPOL0	R/W	capture polarity 0	0
			0 positive edge capture	
			1 negative edge capture	
7:4	reserved	-	reserved	-
3	CAPEN3	R/W	enable capture 3	0
			0 disabled	
			1 enabled	

Table 585. Capture configuration register - Offset: 8h...continued

Bit	Name	Access	Description and value	Reset value
2	CAPEN2	R/W	enable capture 2	0
			0 disabled	
			1 enabled	
1	CAPEN1	R/W	enable capture 1	0
			0 disabled	
			1 enabled	
0	CAPEN0	R/W	enable capture 0	0
			0 disabled	
			1 enabled	

45.4.9.2.4 Capture clear register (CAPCLR)

This read-only register allows clearing previous capture values, allowing new captures to take place.

Table 586. Capture clear register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved	-
3	CAPCLR3	W	clear capture 3 Writing 1 to this bit clears the CAP3 register value.	0
2	CAPCLR2	W	clear capture 2 Writing 1 to this bit clears the CAP2 register value.	0
1	CAPCLR1	W	clear capture 1 Writing 1 to this bit clears the CAP1 register value.	0
0	CAPCLR0	W	clear capture 0 Writing 1 to this bit clears the CAP0 register value.	0

45.4.9.2.5 Capture register (CAP0 - CAP3)

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 587. Offset

Register	Offset
CAP0	10h
CAP1	14h
CAP2	18h
CAP3	1Ch

Table 588. Capture register

Bit	Name	Access	Description and value	Reset value
31	VALID	R	capture valid A value has been captured based on a transition of the related UTICK_CAPn pin. Cleared with writing to the related bit in the CAPCLR register.	-
30:0	CAP_VALUE	R	capture value for the related capture event (UTICK_CAPn) Note: <i>At the moment of the capture event, the value is one lower than the actual value of the micro-tick timer.</i>	0

45.4.10 Analog control (ANALOG_CTRL)

Disclaimer

Do not change the values written to this register. The SDK power library handles these registers. The user must 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.

45.4.10.1 ANACTRL memory map

The ANALOG_CTRL registers are located at base address: 4001_3000h

Table 589. Analog control registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
ANALOG_CTRL_STATUS	R	4h	flash status registers	32		Section 45.4.10.2.1
FREQ_ME_CTRL	R/W	Ch	frequency measure function control register	32	0000_0000h	Section 45.4.10.2.2
FRO192M_CTRL	R/W	10h	192 MHz free running oscillator (FRO) clock control register	32		Section 45.4.10.2.3
FRO192M_STATUS	R	14h	192 MHz free running oscillator (FRO) status register	32		Section 45.4.10.2.4
BLE_STATUS	R	28h	status of the analog components of the Bluetooth Low Energy radio	32		Section 45.4.10.2.5
BLE_CTRL	R/W	2Ch	Bluetooth Low Energy control	32		Section 45.4.10.2.6
BOD_DCDC_INT_CTRL	R/W	30h	brownout detectors (BoDs) and DC-DC interrupts generation control register	32		Section 45.4.10.2.7
BOD_DCDC_INT_STATUS	R	34h	BoDs and DC-DC interrupts status register	32		Section 45.4.10.2.8
RINGO0_CTRL	R/W	40h	first ring oscillator module control register	32		Section 45.4.10.2.9
RINGO1_CTRL	R/W	44h	second ring oscillator module control register	32		Section 45.4.10.2.10
RINGO2_CTRL	R/W	48h	third ring oscillator module control register			Section 45.4.10.2.11

45.4.10.2 Register descriptions

45.4.10.2.1 Flash status registers (ANALOG_CTRL_STATUS)

Table 590. Flash status registers - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:14	reserved	-	reserved	-
31:24	FINAL_TEST_DONE_VECT	R	reserved Indicates current status of final test	-
23:14	reserved	-	reserved Read value is undefined.	0
13	FLASH_INIT_ERROR	R	flash initialization error status 0b no error 1b at least one error occurred during flash initialization	0
12	FLASH_PWRDWN	R	flash power-down status 0b flash is not in power-down mode 1b flash is in power-down mode	0
11:0	reserved	-	reserved Read value is undefined.	-

45.4.10.2.2 Frequency measure function control register (FREQ_ME_CTRL)

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: $F_{\text{target}} = (\text{CAPVAL} * \text{Reference}) / ((2^{\text{SCALE}}) - 1)$. Before starting a frequency measurement by setting the PROG bit in FREQ_ME_CTRL, select the reference and target frequencies using the FREQMEAS_REF and FREQMEAS_TARGET.

Table 591. Frequency measure function control register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31	PROG	R/W	Set this bit to 1 to initiate a frequency measurement cycle. When the measurement cycle has completed and there is valid capture data in the CAPVAL field (bits 30:0), hardware clears this bit.	0
30:0	CAPVAL_SCALE	-	Frequency measure result/Frequency measure scale CAPVAL = FREQ_ME_CTRL[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). Defines the power of the 2-count that reference counter counts to during measurement. Note: The minimum count is 2 ie $2^2 = 4$. The result of freq_me_plus can be calculated as follows: $\text{freq} = (\text{CAPVAL} * (\text{uint64\_t})\text{freq\_refclk}) / ((2^{\text{SCALE}}) - 1)$.	0

45.4.10.2.3 192 MHz free running oscillator (FRO) clock control register (FRO192M_CTRL)

This register is used to enable/disable the different divided clocks from the 192 MHz clock

Table 592. 192 MHz free running oscillator (FRO) clock control register - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:19	reserved	R/W	reserved Read value is undefined, only zero must be written.	0
18	ENA_64MHZCLK	R/W	64 MHz clock control	0
			0b 64 MHz clock is disabled	
			1b 64 MHz clock is enabled	
17	ENA_48MHZCLK	R/W	48 MHz clock control	0
			0b 48 MHz clock is disabled	
			1b 48 MHz clock is enabled	
16	ENA_32MHZCLK	R/W	32 MHz clock control	0
			0b 32 MHz clock is disabled	
			1b 32 MHz clock is enabled	
15	ENA_24MHZCLK	R/W	24 MHz clock control	0
			0b 24 MHz clock is disabled	
			1b 24 MHz clock is enabled	
14	ENA_12MHZCLK	R/W	12 MHz clock control	1
			0b 12 MHz clock is disabled	
			1b 12 MHz clock is enabled	
13:0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.4.10.2.4 192 MHz free running oscillator (FRO) status register (FRO192M_STATUS)

Table 593. 192 MHz free running oscillator (FRO) status register - Offset: 14h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved Read value is undefined.	-
1	ATB_VCTRL	R	CCO threshold voltage detector output (signal vcco_ok) When the CCO voltage crosses the threshold voltage of a SLVT transistor, this output signal goes high. It is also possible to observe the clk_valid signal.	1
0	CLK_VALID	R	output clock valid signal Indicates that the CCO clock has settled.	1
			0b no output clock present None of 12 MHz, 48 MHz, or 96 MHz clocks is available.	
			1b clock is present If they are enabled by FRO192M_CTRL.ENA_12 MHzCLK/ENA_48MHzCLK/ENA_96MHzCLK the 12 MHz, 48 MHz, or 96 MHz clock can be output.	

45.4.10.2.5 Status of the analog components of the Bluetooth Low Energy radio (BLE_STATUS)

Table 594. Status of the analog components of the Bluetooth Low Energy radio - Offset: 28h

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved Read value is undefined.	-
3	BLE_DST_INT_STATUS	R	Bluetooth Low Energy deep-sleep timer interrupt status after interrupt enable	0
			0b no interrupt pending	
			1b interrupt pending	
2	BLE_DST_STATUS	R	Bluetooth Low Energy deep-sleep timer interrupt status before interrupt enable	0
			0b no interrupt pending	
			1b interrupt pending	
1	PLLFLGCLK	R	Bluetooth Low Energy PLL lock indicator output	0
			0b PLL not locked	
			1b PLL locked	
0	DTST_XOCLKRDY	R	Clock ready signal from Bluetooth Low Energy 32 MHz xtal oscillator	0
			0b clock not ready	
			1b clock ready	

45.4.10.2.6 Bluetooth Low Energy control (BLE_CTRL)

Table 595. Bluetooth Low Energy - Offset: 2Ch

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved Read value is undefined.	-
1	BLE_DST_INT_CLR	R	Bluetooth Low Energy deep-sleep timer interrupt clear Write 1 to clear the interrupt.	0
0	BLE_DST_INT_ENABLE	R	Bluetooth Low Energy deep-sleep timer interrupt control	0
			0b Bluetooth Low Energy deep-sleep timer interrupt is disabled	
			1b Bluetooth Low Energy deep-sleep timer interrupt is enabled	

45.4.10.2.7 Brownout detectors (BoDs) and DCDC interrupt generation control register (BOD_DCDC_INT_CTRL)

Table 596. Brownout detectors (BoDs) and DCDC interrupt generation control register - Offset: 30h

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved Read value is undefined. Only zeros must be written	-
5	DCDC_INT_CLEAR	R/W	DCDC interrupt clear 1: Clear the interrupt. Self-cleared bit.	0
4	DCDC_INT_ENABLE	R/W	DCDC interrupt control	0
			0b DCDC interrupt is disabled	
			1b DCDC interrupt is enabled	
3	BOD2_INT_CLEAR	R/W	BOD2 interrupt clear 1: Clear the interrupt. Self-cleared bit.	0
2	BOD2_INT_ENABLE	R/W	BOD2 interrupt control	0
			0b BOD2 interrupt is disabled	
			1b BOD2 interrupt is enabled	
1	BOD1_INT_CLEAR	R/W	BOD1 interrupt clear 1: Clear the interrupt. Self-cleared bit.	0
0	BOD1_INT_ENABLE	R/W	BOD1 interrupt control	0
			0b BOD1 interrupt is disabled	
			1b BOD1 interrupt is enabled	

45.4.10.2.8 BoDs and DCDC interrupt status register (BOD_DCDC_INT_STATUS)

Table 597. BoDs and DCDC interrupt status register - Offset: 34h

Bit	Name	Access	Description and value	Reset value
31:9	reserved	-	reserved Read value is undefined.	-
8	DCDC_VAL	R	Current value of DCDC power status output	1
			0b DCDC output Voltage is below the targeted regulation level.	
			1b DCDC output Voltage is above the targeted regulation level.	
7	DCDC_INT_STATUS	R	DCDC interrupt status after interrupt enable	0
			0b no interrupt pending	
			1b interrupt pending	
6	DCDC_STATUS	R	DCDC interrupt status before interrupt enable	0
			0b no interrupt pending	
			1b interrupt pending	
5	BOD2_VAL	R	Current value of BOD2 power status output	0
			0b VBAT INT voltage level is below the threshold.	
			1b VBAT INT voltage level is above the threshold.	
4	BOD2_INT_STATUS	R	BOD2 interrupt status after interrupt enable	0
			0b no interrupt pending	
			1b interrupt pending	
3	BOD2_STATUS	R	BOD2 interrupt status before interrupt enable	0
			0b no interrupt pending	
			1b interrupt pending	
2	BOD1_VAL	R	Current value of BOD1 power status output	1
			0b VBAT INT voltage level is below the threshold.	
			1b VBAT INT voltage level is above the threshold.	
1	BOD1_INT_STATUS	R	BOD1 interrupt status after interrupt enable	0
			0b no interrupt pending	
			1b interrupt pending	
0	BOD1_STATUS	R	BOD1 interrupt status before interrupt enable	0
			0b no interrupt pending	
			1b interrupt pending	

45.4.10.2.9 First ring oscillator module control register (RINGO0_CTRL)

Table 598. First ring oscillator module control register - Offset: 40h

Bit	Name	Access	Description and value	Reset value
31:13	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
12	E_PN1	R/W	second PN (P-transistor and N-transistor processing) monitor control	0
			0b second PN-based ringo is disabled	
			1b second PN-based ringo is enabled	
11	E_PN0	R/W	first PN (P-transistor and N-transistor processing) monitor control	0
			0b first PN-based ringo is disabled	
			1b first PN-based ringo is enabled	
10	E_IV1	R/W	second inverter-based ringo control	0
			0b second INV-based ringo is disabled	
			1b second INV-based ringo is enabled	
9	E_IV0	R/W	first inverter-based ringo control	0
			0b first INV-based ringo is disabled	
			1b first INV-based ringo is enabled	
8	E_NR1	R/W	second NOR2-based ringo control	0
			0b second NORD2-based ringo is disabled	
			1b second NORD2-based ringo is enabled	
7	E_NR0	R/W	first NOR2-based ringo control	0
			0b first NORD2-based ringo is disabled	
			1b first NORD2-based ringo is enabled	
6	E_ND1	R/W	second NAND2-based ringo control	0
			0b second NAND2-based ringo is disabled	
			1b second NAND2-based ringo is enabled	
5	E_ND0	R/W	first NAND2-based ringo control	0
			0b first NAND2-based ringo is disabled	
			1b first NAND2-based ringo is enabled	
4	PD	R/W	ringo module power control	1
			0b ringo module is enabled	
			1b ringo module is disabled	

Table 598. First ring oscillator module control register - Offset: 40h...continued

Bit	Name	Access	Description and value	Reset value
3:2	SWN_SWP	R/W	PN-ringos (P-transistor and N-transistor processing) control	0
			00b normal mode	
			01b p-monitor mode Measure with weak P transistor.	
			10b p-monitor mode Measure with weak N transistor.	
			11b don't use	
1	FS	R/W	ringo frequency output divider	0
			0b high-frequency output (frequency < 100 MHz)	
			1b low-frequency output (frequency < 10 MHz)	
0	SL	R/W	select short or long ringo (for all ringos types)	0
			0B select short ringo (few elements)	
			1B select longt ringo (many elements)	

45.4.10.2.10 Second ring oscillator module control register (RINGO1_CTRL)

Table 599. Second ring oscillator module control register - Offset: 44h

Bit	Name	Access	Description and value	Reset value
31:13	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
12	E_PN1	R/W	second PN (P-transistor and N-transistor processing) monitor control	0
			0b second PN-based ringo is disabled	
			1b second PN-based ringo is enabled	
11	E_PN0	R/W	first PN (P-transistor and N-transistor processing) monitor control	0
			0b first PN-based ringo is disabled	
			1b first PN-based ringo is enabled	
10	E_IV1	R/W	second inverter-based ringo control	0
			0b second INV-based ringo is disabled	
			1b second INV-based ringo is enabled	
9	E_IV0	R/W	first inverter-based ringo control	0
			0b first INV-based ringo is disabled	
			1b first INV-based ringo is enabled	
8	E_NR1	R/W	second NOR2-based ringo control	0
			0b second NORD2-based ringo is disabled	
			1b second NORD2-based ringo is enabled	

Table 599. Second ring oscillator module control register - Offset: 44h...continued

Bit	Name	Access	Description and value	Reset value
7	E_NR0	R/W	first NOR2-based ringo control	0
			0b first NORD2-based ringo is disabled	
			1b first NORD2-based ringo is enabled	
6	E_ND1	R/W	second NAND2-based ringo control	0
			0b second NAND2-based ringo is disabled	
			1b second NAND2-based ringo is enabled	
5	E_ND0	R/W	first NAND2-based ringo control	0
			0b first NAND2-based ringo is disabled	
			1b first NAND2-based ringo is enabled	
4	PD	R/W	ringo module power control	1
			0b ringo module is enabled	
			1b ringo module is disabled	
3:2	SWN_SWP	R/W	PN-ringos (P-transistor and N-transistor processing) control	0
			00b normal mode	
			01b p-monitor mode Measure with weak P transistor.	
			10b p-monitor mode Measure with weak N transistor.	
			11b don't use	
1	FS	R/W	ringo frequency output divider	0
			0b high-frequency output (frequency < 100 MHz)	
			1b low-frequency output (frequency < 10 MHz)	
0	SL	R/W	select short or long ringo (for all ringos types)	0
			0B select short ringo (few elements)	
			1B select longt ringo (many elements)	

45.4.10.2.11 Third ring oscillator module control register (RINGO2_CTRL)

Table 600. Third ring oscillator module control register - Offset: 48h

Bit	Name	Access	Description and value	Reset value
31:9	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
8	E_M5	R/W	metal 5 (M5) monitor control	0
			0b ringo is disabled	
			1b ringo is enabled	
7	E_M4	R/W	metal 4 (M4) monitor control	0
			0b ringo is disabled	
			1b ringo is enabled	
6	E_M3	R/W	metal 3 (M3) monitor control	0
			0b ringo is disabled	
			1b ringo is enabled	
5	E_M2	R/W	metal 2 (M2) monitor control	0
			0b ringo is disabled	
			1b ringo is enabled	
4	E_R35	R/W	-	0
			0b ringo is disabled	
			1b ringo is enabled	
3	E_R24	R/W	-	0
			0b ringo is disabled	
			1b ringo is enabled	
2	PD	R/W	ringo module power control	1
			0b ringo module is enabled	
			1b ringo module is disabled	
1	FS	R/W	ringo frequency output divider	0
			0b high-frequency output (frequency < 100 MHz)	
			1b low-frequency output (frequency < 10 MHz)	
0	SL	R/W	select short or long ringo (for all ringos types)	0
			0b select short ringo (few elements)	
			1b select longt ringo (many elements)	

45.4.11 Power management control (PMC)

45.4.11.1 PMC memory map

The PMC registers are located at base address: 4002_0000h

Table 601. Power management control registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CTRL	R/W	0h	power management control	32		Section 45.4.11.2.1
STATUS	R	4h	power management controller FSM status	32		Section 45.4.11.2.2
RESETCTRL	R/W	8h	reset control	32		Section 45.4.11.2.3
RESETCAUSE	R/W	Ch	reset cause register	32		Section 45.4.11.2.4
DCDC0	R/W	10h	DCDC control register	32		Section 45.4.11.2.5
BIAS	R/W	18h	bias current source control register	32		Section 45.4.11.2.6
LDOMEM	R/W	20h	control register for LDO which supplies memory power domain	32		Section 45.4.11.2.7
LDODIG	R/W	24h	control register for LDO which supplies digital power domain	32		Section 45.4.11.2.8
BOD1	R/W	30h	control register for brownout detector 1 module	32		Section 45.4.11.2.9
BOD2	R/W	34h	control register for brownout detector 2 module	32		Section 45.4.11.2.10
FROM192M	R/W	40h	192 MHz FRO control register	32		Section 45.4.11.2.11
FRO1M	R/W	44h	1 MHz FRO control register	32		Section 45.4.11.2.12
FRO32K	R/W	48h	32.768 kHz FRO control register	32		Section 45.4.11.2.13
XTAL32K	R/W	4Ch	32.768 kHz crystal oscillator control register	32		Section 45.4.11.2.14
XTALBLE	R/W	54h	Bluetooth Low Energy XTAL overrule	32		Section 45.4.11.2.15
WAKEUIOCTRL	R/W	64h	WAKEUP IO configuration register	32		Section 45.4.11.2.16
WAKEIOCAUSE	W	68h	wake-up I/O source from deep power-down mode	32		Section 45.4.11.2.17
STATUSPWR	R	70h	status register for DCDC and LDO after power-up	32		Section 45.4.11.2.18

Table 601. Power management control registers...continued

Name	Access	Offset	Description	Width (bits)	Reset value	Section
STATUSCLK	R	74h	FRO and XTAL status register	32		Section 45.4.11.2.19
AOREG0	R/W	80h	write-once general purpose always on domain data storage	32	0000_0000h	Section 45.4.11.2.20
AOREG1	R/W	84h	general purpose always on domain data storage	32	0000_0000h	Section 45.4.11.2.21
AOREG2	R/W	88h	general purpose always on domain data storage	32	0000_0000h	Section 45.4.11.2.22
RTCOSC32K	R/W	98h	RTC 1 KHZ and 1 Hz clocks source control register	32		Section 45.4.11.2.23
OSTIMER	R/W	9Ch	OS timer control register	32		Section 45.4.11.2.24
TIMEOUTEVENTS	R/W	A4h	timeout status events during IC power-up	32		Section 45.4.11.2.25
TIMEOUT	R/W	A8h	Bluetooth Low Energy restart timeout configuration register	32		Section 45.4.11.2.26
PDSLEEPCFG0	R/W	B0h	power-down configuration during low-power mode	32		Section 45.4.11.2.27
SRAMRETCTRL	R/W	B4h	configuration register for SRAM retention during low-power modes	32		Section 45.4.11.2.28
PDRUNCFG0	R/W	B8h	power-down configuration during active mode	32		Section 45.4.11.2.29
PDRUNCFGSET0	W	C0h	power-down configuration during active mode set register	32	0000_0000h	Section 45.4.11.2.30
PDRUNCFGCLR0	W	C8h	power-down configuration during active mode clear register	32	0000_0000h	Section 45.4.11.2.31
ROMCTRL	R/W	D0h	ROM control	32		Section 45.4.11.2.32
SRAMCTRL	R/W	D4h	SRAM control	32		Section 45.4.11.2.33
SRAMCTRL0	R/W	D8h	RAMX0 to RAM20 power modes controls	32		Section 45.4.11.2.34
SRAMCTRL1	R/W	DCh	Bluetooth Low Energy RAM power mode controls	32		Section 45.4.11.2.35
SRAMASSISTCTRL0	R/W	E0h	control register for SRAM read/write assist margin test	32		Section 45.4.11.2.36
SRAMASSISTCTRL1	R/W	E4h	control register for Bluetooth Low Energy SRAM read/write assist margin test	32		Section 45.4.11.2.37

Table 601. Power management control registers...continued

Name	Access	Offset	Description	Width (bits)	Reset value	Section
DFTENDIS	R/W	F8h	control register for Bluetooth Low Energy SRAM read/write assist margin test	32		Section 45.4.11.2.38
DFTDCDC	R/W	FCh	control register to enable/disable items during DfT.	32		Section 45.4.11.2.39
DFTCAPTEST	R/W	100h	control register for COT and XO32K capacitor bank settings during DfT mode	32		Section 45.4.11.2.40

45.4.11.2 Register descriptions

45.4.11.2.1 Power management control (CTRL)

Table 602. Power management control - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
4	POFFMODE	R/W	power-off mode control	0
			0b power mode defined by LPMODE	
			1b power off	
3:2	SELCLOCK	R/W	select power management controller (PMC) functional clock	0
			00b 1 MHz free running oscillator	
			01b 12 MHz free running oscillator	
			10b 32.768 kHz RTC clock	
1:0	LPMODE	R/W	power mode control	0
			00b ACTIVE power mode	
			01b DEEP SLEEP low-power mode	
			10b POWER DOWN low-power mode	
			11b DEEP POWER DOWN low-power mode	

45.4.11.2.2 Power management controller FSM status (STATUS)

Table 603. Power management controller FSM status - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:25	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
24:20	FSMBLESTATE	R	Bluetooth Low Energy finite state machine (FSM) status 0_0000b reset 0_0001b link layer and memory enable 0_0010b enable clock out 0_0011b enable oscillator 0_0100b low-power mode - link-layer deep sleep and phy shutdown 0_0101b low-power mode disable oscillator 0_0110b radio active 0_0111b low-power mode assert isolations and resets 0_1000b power-off mode - trigger hresetn 0_1001b power-off mode - radio restart 0_1010b power-off mode - remove power 0_1011b power-off mode - radio off 0_1100b low-power mode - radio restart 0_1101b low-power mode - enable oscillator 0_1110b power-off mode - assert isolation and resets 0_1111b low-power mode - enable clock output 1_1000b power-off mode - enable oscillator 1_1001b power-off mode - enable clock out	0
19:18	BOOTMODE	R	latest IC boot cause 00b Latest IC boot was a full-power cycle boot sequence (PoR, pin reset, brownout 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.	0

Table 603. Power management controller FSM status - Offset: 4h...continued

Bit	Name	Access	Description and value	Reset value
17:15	FSMDPWD	R	DEEP POWER DOWN finite state machine (FSM) status	0
			000b INIT The IC is entering deep power down - initialization.	
			001b POWER_SHUTOFF The IC is entering deep power down - power shutoff.	
			010b DONE The IC is entering deep power down - done.	
			011b DISABLE_FRO The IC is entering deep power down - disable FRO.	
14:11	FSMPWDN	R	POWER DOWN finite state machine (FSM) status	0
			0000b IDLE The IC is entering power down - idle state.	
			0001b DISABLE The IC is entering power down - disable.	
			0010b POWER_SHUTOFF The IC is entering power down - power shutoff.	
			0011b SHUTOFF The IC is entering power down - shutoff.	
			0100b PWRUP_ENABLE_DCDC The IC is exiting power down - enable DCDC.	
			0101b PWRUP_BG The IC is exiting power down - power-up bandgap.	
			0110b DISABLE_FRO The IC is entering power down - disable FRO.	
			0111b DONE The IC is entering power down - done.	
			1010b PWRUP_INIT_FLASH The IC is exiting from power down - power-up flash initialization.	
			1011b PWRUP_DONE The IC is exiting deep sleep - power-up done.	
			1100b PWRUP_ENABLE_LDO The IC is exiting power down - enable LDO.	
			1101b PWRUP_SWITCH_DOMAINS The IC is exiting power down - power-up switch domains.	
			1110b PWRUP_RELEASE_ISOLATION The IC is exiting power down - power-up release isolation.	
			1111b PWRUP_WAIT_FRO192M The IC is exiting power down - power-up wait for FRO192M.	

Table 603. Power management controller FSM status - Offset: 4h...continued

Bit	Name	Access	Description and value	Reset value
10:7	FSMDSLP	R	DEEP SLEEP finite state machine (FSM) status	0
			0000b IDLE The IC is entering deep sleep - idle state.	
			0001b PREPARE The IC is entering deep sleep - prepare.	
			0010b DISABLE_FRO The IC is entering deep sleep - disable FRO.	
			0011b SWITCH_TO_DEEPSLEEP The IC is entering deep sleep - switch to deep sleep.	
			0100b PWRUP_ENABLE_LDO The IC is exiting deep sleep - enable LDO.	
			0101b PWRUP_ENABLE_DCDC The IC is exiting deep sleep - enable DCDC.	
			0110b DONE The IC is entering deep sleep - done.	
			0111b PWRUP_BG The IC is exiting deep sleep - wait on bandgap.	
			1010b PWRUP_DONE The IC is exiting deep sleep - power-up done.	
			1100b SWITCH_DOMAINS The IC is exiting deep sleep - switch domains.	
			1101b ENABLE_FLASH The IC is exiting deep sleep - enable flash.	
			1110b ACTIVATE_SRAM The IC is exiting deep sleep - activate SRAM.	
			1111b PWUP_WAIT_FRO192M The IC is exiting deep sleep - wait on FRO192M.	
6:3	FSMPWUP	R	POWER UP finite state machine (FSM) status	0
			0000b RESET The IC is powering up - reset state.	
			0001b ENABLE_DCDC The IC is powering up - enable DCDC.	
			0010b SWITCH_DOMAINS The IC is powering up - switch domains.	
			0011b ENABLE_LDO The IC is powering up - enable LDO.	
			0100b ACTIVATE_MEM The IC is powering up - activate memories.	
			0101b RELEASE_ISOLATION The IC is powering up - release isolations.	

Table 603. Power management controller FSM status - Offset: 4h...continued

Bit	Name	Access	Description and value		Reset value
6:3 (cont'd)	FSMPWUP	R	0110b	ENABLE_FLASH The IC is powering up - enable flash.	0
			0111b	WAIT_FRO192M The IC is powering up - wait on FRO192M	
			1010b	DONE The IC is powering up - done.	
			1100b	RELEASE_RESET The IC is powering up - release reset.	
			1101b	TRANSFER_TRIM The IC is powering up - transfer trim.	
			1110b	SWITCH_CLOCK The IC is powering up - switch clocks.	
			1111b	RELEASE_CPU The IC is powering up - release CPU	
2:0	FSMMAIN	R	power management controller main finite state machine (FSM) status		0
			000b	POWER UP The IC is powering up.	
			001b	ACTIVE Power up is completed. The IC is in normal functional operation mode.	
			010b	DEEP SLEEP The IC has entered DEEP SLEEP mode.	
			011b	SLEEP The IC has entered SLEEP mode.	
			100b	IC structural TEST mode The IC has entered in IC Test mode.	
			101b	POWEROFF The IC has entered POWEROFF mode.	
			110b	POWER DOWN The IC has entered POWER DOWN mode.	
			111b	DEEP POWER DOWN The IC has entered DEEP POWER DOWN mode.	

45.4.11.2.3 Reset control (RESETCTRL)

Table 604. Reset control - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:30	BOD2RESETENA_SECURE_DP	R/W	BOD2 reset enable	0
			01b any other value than b10; BOD2 reset is enable	
			10b BOD2 reset is disable	
29:28	BOD1RESETENA_SECURE_DP	R/W	BOD1 reset enable	0
			01b any other value than b10; BOD1 reset is enable	
			10b BOD1 reset is disable	
27:9	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
8	CRCERRORRESETENABLE	R/W	CPU retention CRC error reset enable	0
			0b CPU retention CRC error reset is disable	
			1b CPU retention CRC error reset is enable	
7:6	BOD2RESETENA_SECURE	R/W	BOD2 reset enable	0
			01b any other value than b10; BOD2 reset is enable	
			10b BOD2 reset is disable	
5:4	BOD1RESETENA_SECURE	R/W	BOD1 reset enable	0
			01b any other value than b10; BOD1 reset is enable	
			10b BOD1 reset is disable	
3	SWRRESETENABLE	R/W	software reset enable	0
			0b software reset is disable	
			1b software reset is enable	
2:1	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
0	DPDWAKEUPRESETENABLE	R/W	wake-up from DEEP POWER DOWN reset event (either from wake up I/O or RTC enable)	0
			0b reset event from DEEP POWER DOWN mode is disable	
			1b reset event from DEEP POWER DOWN mode is enable	

45.4.11.2.4 Reset cause register (RESETCAUSE)

Table 605. Reset cause register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:12	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
11	CRCERROR	R/W	1: The last chip reset was caused by a CRC error during CPU state retention restore. Write '1' to clear this bit.	0
10	BLEWUP	R/W	1: The last chip wake-up was caused by the Bluetooth Low Energy timer wake-up. Write '1' to clear this bit.	0
9	CDOGRESET	R/W	1: The last chip reset was caused by the code watchdog. Write '1' to clear this bit.	0
8	DPDRESET_OSTIMER	R/W	1: The last chip reset was caused by OS Event Timer reset event during DEEP POWER DOWN mode. Write '1' to clear this bit.	0
7	DPDRESET_RTC	R/W	1: The last chip reset was caused by RTC. Write '1' to clear this bit.	0
6	DPDRESET_WAKEUIO	R/W	1: The last chip reset was caused by a wake-up I/O reset event during DEEP POWER DOWN mode. Write '1' to clear this bit.	0
5	SWRRESET	R/W	1: The last chip reset was caused by a software reset. Write '1' to clear this bit.	0
4	WDTRESET	R/W	1: The last chip reset was caused by the watchdog timer. Write '1' to clear this bit.	0
3	SYSTEMRESET	R/W	1: The last chip reset was caused by a system reset requested by the Arm CPU. Write '1' to clear this bit.	0
2	BODRESET	R/W	1: The last chip reset was caused by a brownout detector, either BoD1 or BoD2. Write '1' to clear this bit.	0
1	PADRESET	R/W	1: The last chip reset was caused by a pin reset. Write '1' to clear this bit.	0
0	POR	R/W	1: The last chip reset was caused by a power-on reset. Write '1' to clear this bit.	

45.4.11.2.5 DCDC control register (DCDC0)

Table 606. DCDC control register - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
22	ROBUST_STARTUP_DISABLE	R/W	block BOD-controlled DCDC buck and boost control	0
21	ENABLE_BYPASS	R/W	enable low-power bypass switch	0
20:18	TRIMPZCD	R/W	trim bits P-ZCD	0
17:15	TRIMNZCD	R/W	trim bits N-ZCD	0
14	PASSTHROUGH_EN	R/W	enable pass through	1
			0 disabled	
			1 enabled	
13:12	INDUCTOR	R/W	scale current mirror	0
			00b 1.0 μ H	
			01b 2.2 μ H	
			10b 4.7 μ H	
11:8	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
7:4	RC	R/W	calibration RC time for buck mode	0
3:0	VOLTAGE	R/W	DCDC output voltage level - BUCK mode: $V_{BAT\_LV} = (VOLTAGE[2:0] \times 0.1 + 1.1) \text{ V}$ - BOOST mode: $V_{BAT\_HV} = (VOLTAGE[3:0] \times 0.1 + 1.7) \text{ V}$	0

45.4.11.2.6 Bias current source control register (BIAS)

Table 607. Bias current source control register - Offset: 18h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
15:12	LPDCSHIFT	R/W	trimming bits to adjust DC shift in low-power mode	0
11:7	IREFTRIM	R/W	bias current trimming	0
6:5	CURVE	R/W	trimming bits to adjust deviations in curvature on silicon	0
4:0	DCSHIFT	R/W	trimming bits to adjust DC shift in active mode	0

45.4.11.2.7 LDOMEM (LDOMEM)

Table 608. LDOMEM - Offset: 20h

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
4	VADJ_PWD	R/W	sets the LDO output level to 0.63 V or 0.66 V in low-power mode Note: The PDSLEEPCFG.PDEN_BIAS[0] value determines the voltage level in low-power mode.	0
3:0	VADJ	R/W	sets the LDO output level (4 bit, only 2 used)	0
			0000 0.9 V	
			0001 0.95 V	
			0010 1.0 V	
			0011 1.05 V	

45.4.11.2.8 LDODIG (LDODIG)

Table 609. LDODIG - Offset: 24h

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
4	VADJ_PWD	R/W	sets the LDO output level to 0.63 V or 0.66 V in low-power mode Note: The PDSLEEPCFG.PDEN_BIAS[0] value value determines the voltage level in low-power mode.	0
3:0	VADJ	R/W	sets the LDO output level (4 bit, only 2 used)	0
			0000 0.9 V	
			0001 0.95 V	
			0010 1.0 V	
			0011 1.05 V	

45.4.11.2.9 BOD1 (BOD1)

Table 610. BOD1 - Offset: 30h

Bit	Name	Access	Description and value	Reset value
31:9	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
8	LVLSEL	R/W	BOD level select when mode is set to zero	0
			0b reference is sum of threshold plus hysteresis	
			1b reference is threshold	

Table 610. BOD1 - Offset: 30h...continued

Bit	Name	Access	Description and value	Reset value
7	MODE	R/W	BOD operation mode select	1
			0b level detector mode	
			1b BOD mode	
6:5	HYST	R/W	BOD hysteresis control	0
			00b 25 mV	
			01b 50 mV	
			10b 75 mV	
			11b 100 mV	
4:0	TRIGLVL	R/W	BOD trigger level	0x4
			0_0000b - 0_1010b Between 1.00 V and 1.50 V, the increment is in steps of 0.05 V	
			0_1010b - 1_1010b Between 1.50 V and 3.10 V, the increment is in steps of 0.10 V	

45.4.11.2.10 BOD2 (BOD2)

Table 611. BOD2 - Offset: 34h

Bit	Name	Access	Description and value	Reset value
31:9	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
8	LVLSEL	R/W	BOD level select when mode is set to zero	0
			0b reference is sum of threshold plus hysteresis	
			1b reference is threshold	
7	MODE	R/W	BOD operation mode select	1
			0b level detector mode	
			1b BOD mode	
6:5	HYST	R/W	BOD hysteresis control	3
			00b 25 mV	
			01b 50 mV	
			10b 75 mV	
			11b 100 mV	

Table 611. BOD2 - Offset: 34h...continued

Bit	Name	Access	Description and value	Reset value
4:0	TRIGLVL	R/W	BOD trigger level	0x13
			0_0000b - 0_1010b	
			Between 1.00 V and 1.50 V, the increment is in steps of 0.05 V	
			0_1010b - 1_1010b	
			Between 1.50 V and 3.10 V, the increment is in steps of 0.10 V	

45.4.11.2.11 FRO192M control (FRO192M)

Table 612. FRO192M control - Offset: 40h

Bit	Name	Access	Description and value	Reset value
31:20	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
19:14	TC	R/W	temperature compensation	1
13:6	CURDAC	R/W	fine frequency trimming	0
5:0	BIAS	R/W	coarse frequency trimming	0

45.4.11.2.12 FRO1M control (FRO1M)

Table 613. FRO1M control - Offset: 44h

Bit	Name	Access	Description and value	Reset value
31:14	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
13:9	DIVSEL	R/W	divider selection bits	00
			0_0000b - 1_1110b	
			Between 2.0 bits and 62.0 bits, the increment is in steps of 2.	
			1_1111b 1.0	
8:7	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
6:0	FREQSEL	R/W	frequency trimming bits	0

45.4.11.2.13 FRO32K control (FRO32K)

Table 614. FRO32K control - Offset: 48h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
15:7	CAPCAL	R/W	capacitive DAC calibration bits (fine frequency trimming).	0

Table 614. FRO32K control - Offset: 48h...continued

Bit	Name	Access	Description and value	Reset value
6:4	PTAT	R/W	bias trimming bits (coarse frequency trimming)	0
3:1	NTAT	R/W	temperature coefficient trimming bits	0
0	reserved	-	reserved Read value is undefined. Only zeros must be written.	-

45.4.11.2.14 32 KHz crystal oscillator control register (XTAL32K)

Table 615. 32 KHz crystal oscillator control register - Offset: 4Ch

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
15:12	CAP_OSCOUT	R/W	osc_out capacitor bank selection inputs	0
11:8	CAP_OSCIN	R/W	osc_in capacitor bank selection inputs	0
7:6	AMPL	R/W	oscillator amplitude selection inputs	0
5:4	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
3	XTM	R/W	oscillator bypass mode	0
2:1	GM	R/W	reference output current selection inputs	0
0	reserved	-	reserved Read value is undefined. Only zeros must be written.	-

45.4.11.2.15 Bluetooth Low Energy xtal overrule (XTALBLE)

Table 616. Bluetooth Low Energy xtal overrule - Offset: 54h

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
0	CKO_UNGATE_OVERRULE	R/W	overrule the CKO ungate control of the Bluetooth Low Energy crystal	0
			0b crystal oscillator output of the Bluetooth Low Energy is enabled when Bluetooth Low Energy is active	
			1b crystal oscillator output of the Bluetooth Low Energy is disabled	

45.4.11.2.16 Wake-up pin control (WAKEUIOCTRL)

Table 617. Wake-up pin control - Offset: 64h

Bit	Name	Access	Description and value	Reset value
31:22	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
21	WAKEUIO_RSTN	R/W	WAKEUP IO event detector reset control When this bit is set, the RESETCAUSE register is updated when the wake-up IO causes a wake-up event.	0
			0b bloc is reset	
			1b bloc is not reset	
20:11	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
10	LVL	R/W	select wake-up pin event level in power-off	1
			0b wake-up on low level	
			1b wake-up on high level	
9	PDN_EN	R/W	enable/disable wake-up pin pull-down in power-off mode	0
			0b disable pull-down	
			1b enable pull-down	
8	PUP_EN	R/W	enable/disable wake-up pin pull-up in power-off mode	0
			0b disable pull-up	
			1b enable pull-up	
7:1	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
0	ENABLE	R/W	enable/disable wake-up on wake-up pin in deep power-down modes	1

45.4.11.2.17 Wake-up I/O source from deep power-down mode (WAKEIOCAUSE)

This register is relevant when bit DPDRESET_WAKEUIO from register RESETCAUSE is set to '1'.

Table 618. Wake-up I/O source from deep power-down mode - Offset: 68h

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
0	WAKEUP0	W	Allows identification of wake-up I/O 0 as the wake-up source from deep power-down mode	0
			0b Last wake up from deep power-down mode was NOT triggered by wake-up I/O 0.	
			1b Last wake up from deep power-down mode was triggered by wake-up I/O 0.	

45.4.11.2.18 Status power (STATUSPWR)

Table 619. Status power - Offset: 70h

Bit	Name	Access	Description and value	Reset value
31:21	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
20:5	DURATION	R	DCDC start-up duration in clock cycles (based on selected PMC clock) Counter saturates at FFFF.	0
4	LDOEMOK	R	LDOEM power OK	0
3	LDODIGOK	R	LDODIG power OK	0
2	DCDCOK	R	DCDC power OK	0
1:0	DCDCMODE	R	DCDC mode	0
			01 buck mode	
			10 boost mode	

45.4.11.2.19 FRO and xtal status register (STATUSCLK)

Table 620. FRO and xtal status register - Offset: 74h

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
2	XTAL32 KOSCFailure	R	XTAL 32 KHZ oscillator oscillation failure detection indicator	0
		R	0b No oscillation failure has been detected since the last time this bit was cleared.	
		R	1b At least one oscillation failure has been detected since the last time this bit was cleared.	
1	FRO1MCLKVALID		FRO 1 MHz CCO voltage detector output	1

Table 620. FRO and xtal status register - Offset: 74h...continued

Bit	Name	Access	Description and value	Reset value
0	XTAL32KOK		xtal oscillator 32 K OK signal Note: Bit not working as expected (see artf872570).	0
0	reserved	-	reserved Read value is undefined. Only zeroes must be written	-

45.4.11.2.20 Write-once general purpose always on domain data storage (AOREG0)

Table 621. Write-once general purpose always on domain data storage - Offset: 80h

Bit	Name	Access	Description and value	Reset value
31:0	DATA31_0	R/W	general purpose always on domain data storage	0

45.4.11.2.21 General purpose always on domain data storage (AOREG1)

Table 622. General purpose always on domain data storage - Offset: 84h

Bit	Name	Access	Description and value	Reset value
31:0	DATA31_0	R/W	Write-once general purpose always on domain data storage (reset by: PoR, brownout detectors reset) 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.	0

45.4.11.2.22 General purpose always on domain data storage (AOREG2)

Table 623. General purpose always on domain data storage - Offset: 88h

Bit	Name	Access	Description and value	Reset value
31:0	DATA31_0	R/W	general purpose always on domain data storage The ROM API uses this register.	0

45.4.11.2.23 RTC 1 KHZ and 1 Hz clocks source control register (RTCOSC32K)

Table 624. RTC 1 KHZ and 1 Hz clocks source control register - Offset: 98h

Bit	Name	Access	Description and value	Reset value
31	CLK1HZDIVUPDATEREQ	R/W	RTC 1 Hz divider status flag	0
30	CLK1HZDIVHALT	R/W	halts the divider counter	0
29:27	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
26:16	CLK1HZDIV	R/W	Actual division ratio is: 31744 + CLK1HZDIV.	0x3FF

Table 624. RTC 1 KHZ and 1 Hz clocks source control register - Offset: 98h...continued

Bit	Name	Access	Description and value	Reset value
15	CLK1KHZDIVUPDATEREQ	R/W	RTC 1 KHz clock divider status flag	0
14:4	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
3:1	CLK1KHZDIV	R/W	configures the divider to derive the 1 kHz RTC clock from the selected RTC input clock The actual division ratio is 29 + CLK1KHZDIV	4
0	SEL	R/W	selects either the XTAL32K or the FRO32K as the 32.768 kHz clock source for the whole system	0
			0b FRO 32.768 KHz	
			1b XTAL 32.768 KHz	

45.4.11.2.24 OS timer control register (OSTIMER)

Table 625. OS timer control register - Offset: 9Ch

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
5:4	OSTIMERCLKSEL	R/W	OS event timer clock select	0
			00b 32.768 kHz clock FRO32K or XTAL32K (determined by the RTCOSC32K.SEL bit).	
			01b FRO 1 MHz clock	
			10b reserved Do not select this value.	
			11b no clock	
3	OSC_PD	R/W	power-down oscillator 32.768 kHz Either FRO32K or XTAL32K according to RTCOSC32K.SEL.	1
2	DPDWAKEUPENABLE	R/W	wake-up enable in deep power-down mode (to be used in enable deep power-down mode)	0
1	CLOCKENABLE	R/W	enable OS event timer clock	0
0	SOFTRESET	R/W	active high reset	0

45.4.11.2.25 Timeout status events during IC power-up (TIMEOUTEVENTS)

Table 626. Timeout status events during IC power-up - Offset: A4h

Bit	Name	Access	Description and value	Reset value
31:21	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
20	PWDN_BLE_STARTUP	R/W	1: A timeout event occurred during power up from power-down when waiting for Bluetooth Low Energy.	
19	DSLP_BLE_STARTUP	R/W	1: A timeout event occurred during power up from deep sleep when waiting for Bluetooth Low Energy.	
18	PWUP_BLE_STARTUP	R/W	1: A timeout event occurred during power-up when waiting for Bluetooth Low Energy.	
17	PWDN_CLK192MHZ_OK	R/W	1: A timeout event occurred during power-up from the power-down state when waiting for oscillator start-up.	0
16	DSLP_CLK192MHZ_OK	R/W	1: A timeout event occurred during power-up from the low-power state when waiting for oscillator start-up.	0
15	PWUP_CLK192MHZ_OK	R/W	1: A timeout event occurred during power-up when waiting for oscillator start-up.	0
14	PDWN_BLE_SHUTDOWN	R/W	1: A timeout event occurred during power-down when waiting for Bluetooth Low Energy power-down acknowledge.	0
13	DSLP_BLE_SHUTDOWN	R/W	1: A timeout event occurred during deep sleep when waiting for Bluetooth Low Energy power-down acknowledge.	0
12	PDWN_FLASHINIT_DONE	R/W	1: A timeout event occurred during power down when waiting for flash initialization to become functional.	0
11:10	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
9	PDWN_DCDC_OK	R/W	1: A timeout event occurred during power down when waiting for DCDC to become functional.	0
8	PDWN_SRAM_OFF	R/W	1: A timeout event occurred during power down when waiting for SRAM shut off.	0
7	DSLP_SRAM_WAKEUP	R/W	1: A timeout event occurred during deep sleep when waiting for SRAM to become functional.	0
6	DSLP_FLASHINIT_DONE	R/W	1: A timeout event occurred during deep sleep when waiting for flash initialization.	0
5	DSLP_DCDC_OK	R/W	1: A timeout event occurred during deep sleep when waiting for DCDC to become functional.	0
4	DSLP_SRAM_OFF	R/W	1: A timeout event occurred during deep sleep when waiting for SRAM shut off.	0

Table 626. Timeout status events during IC power-up - Offset: A4h...continued

Bit	Name	Access	Description and value	Reset value
3	PWUP_FLASHINIT_DONE	R/W	1: A timeout event occurred during power-up when waiting for flash initialization.	0
2	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
1	PWUP_LDO_OK	R/W	1: A timeout event occurred during power-up when waiting for LDO_MEM or LDO_DIG to become functional.	0
0	PWUP_DCDC_OK	R/W	1: A timeout event occurred during power-up when waiting for DCDC to become functional.	0

45.4.11.2.26 Timeout (TIMEOUT)

Table 627. Timeout - Offset: A8h

Bit	Name	Access	Description and value	Reset value
31:20	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
19:16	BLE_RESTART_OSCEN_TIME_OUT	R/W	timeout value for Bluetooth Low Energy restart Resolution is 32 μ s. Default is 160 μ s.	0
15:0	reserved	-	reserved Read value is undefined. Only zeros must be written.	-

45.4.11.2.27 Power-down configuration during low-power mode (PDSLEEPCFG0)

Table 628. Power-down configuration during low-power mode - Offset: B0h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
22	PDEN_RNG	R/W	enable power down of RNG block in low-power mode	0
			0b RNG module is powered on during low-power mode.	
			1b RNG module is powered off during low-power mode.	
21	reserved	-	reserved Read value is undefined. Only zeros must be written.	-

Table 628. Power-down configuration during low-power mode - Offset: B0h...continued

Bit	Name	Access	Description and value	Reset value
20	PDEN_BLE_XO32M	R/W	enable power down of Bluetooth Low Energy 32 MHz XO in low-power mode	0
			0b 32 MHz crystal oscillator is powered on during low-power mode.	
			1b 32 MHz crystal oscillator is powered off during low-power mode.	
19:18	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
17	PDEN_LDODIG	R/W	enable power down of LDO_DIG in low-power mode	0
			0b LDO DIG power domain is powered on during low-power mode.	
			1b LDO DIG power domain is powered off during low-power mode.	
16	PDEN_LDOMEM	R/W	enable power down of LDO MEM in low-power mode	0
			0b LDO MEM power domain is powered on during low-power mode.	
			1b LDO MEM power domain is powered off during low-power mode.	
15:14	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
13	PDEN_CORE	R/W	enable power down of core domain in deep sleep	0
			0b Digital core power domain is powered on during deep-sleep mode.	
			1b Digital core power domain is powered off during deep-sleep mode.	
12:11	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
10	PDEN_BLE	R/W	enable power down of Bluetooth Low Energy retention domain in low-power mode	0
			0b Bluetooth Low Energy retention domain is powered on during low-power mode.	
			1b Bluetooth Low Energy retention domain is powered off during low-power mode.	
9	reserved	-	reserved Read value is undefined. Only zeros must be written.	-

Table 628. Power-down configuration during low-power mode - Offset: B0h...continued

Bit	Name	Access	Description and value	Reset value
8	PDEN_FRO1M	R/W	controls power during DEEP SLEEP and POWER DOWN FRO1M is always shut down during DEEP POWER DOWN and POWEROFF	0
			0b FRO 1 MHz is powered on during low-power mode.	
			1b FRO 1 MHz is powered off during low-power mode.	
7	PDEN_XTAL32K	R/W	controls power of 32.768 kHz crystal oscillator during DEEP SLEEP, POWER DOWN, and DEEP POWER DOWN XTAL32K is always shut down during POWEROFF.	1
			0b Crystal 32.768 KHz is powered on during low-power mode.	
			1b Crystal 32.768 KHz is powered off during low-power mode.	
6	PDEN_FRO32K	R/W	controls power of 32.768 kHz FRO during DEEP SLEEP, POWER DOWN, and DEEP POWER DOWN FRO32K is always shut down during POWEROFF.	1
			0b FRO 32.768 KHz is powered on during low-power mode.	
			1b FRO 32.768 KHz is powered off during low-power mode.	
5	PDEN_FRO192M	R/W	controls the 192 MHz free running oscillator power during DEEP SLEEP FRO192M is always shut down during POWER DOWN, DEEP POWER DOWN, and POWEROFF.	0
			0b FRO 192 MHz is powered on during low-power mode.	
			1b FRO 192 MHz is powered off during low-power mode.	
4	PDEN_BOD2	R/W	controls BOD2 power during low-power mode BOD2 is always shut down during POWEROFF.	0
			0b BOD2 is powered on during low-power mode.	
			1b BOD2 is powered off during low-power mode.	

Table 628. Power-down configuration during low-power mode - Offset: B0h...continued

Bit	Name	Access	Description and value	Reset value
3	PDEN_BOD1	R/W	controls BOD1 power during low-power mode BOD1 is always shut down during POWEROFF.	0
			0b BOD1 is powered on during low-power mode.	
			1b BOD1 is powered off during low-power mode.	
2:1	PDEN_BIAS	R/W	enable or disable bandgap in high-power mode or low-power mode	0
			00b bias 0.66 V low-power mode enabled Intended to be used for retention mode of memory or digital.	
			01b bias 0.63 V high accuracy mode enabled During active mode, the bias must be set in high accuracy mode	
			1xb bias disabled	
0	PDEN_DCDC	R/W	Controls DCDC power during low-power mode DCDC is always shut down during POWEROFF.	0
			0b DCDC is powered on during low-power mode.	
			1b DCDC is powered off during low-power mode.	

45.4.11.2.28 SRAM retention control (SRAMRETCTRL)

Table 629. SRAM retention control - Offset: B4h

Bit	Name	Access	Description and value	Reset value
31:9	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
8	RETEN_RAMBLESEQL	R/W	controls Bluetooth Low Energy sequencer/list RAM power-down modes during low-power modes	0
			0b DEEP SLEEP: The SRAM instance keeps the configuration it has before entering DEEP SLEEP, whatever it is (normal, light sleep, deep-sleep mode and shut down modes). POWER DOWN, DEEP POWER DOWN, and POWEREDOFF: The SRAM instance is in shutdown mode (in this mode there is no data retention).	
			1b The SRAM enters DEEP SLEEP mode during DEEP SLEEP and POWER DOWN state (in this mode there is data retention), except when it was already in SHUTDOWN state during ACTIVE mode. If it was already in SHUTDOWN during ACTIVE mode, the SRAM remains in SHUTDOWN.	
7	RETEN_RAMBLEDATA	R/W	controls Bluetooth Low Energy data RAM power-down modes during low-power modes	0
			0b DEEP SLEEP: The SRAM instance keeps the configuration it has before entering DEEP SLEEP, whatever it is (normal, light sleep, deep-sleep mode and shut down modes). POWER DOWN, DEEP POWER DOWN, and POWEREDOFF: The SRAM instance is in shutdown mode (in this mode there is no data retention).	
			1b The SRAM enters DEEP SLEEP mode during DEEP SLEEP and POWER DOWN state (in this mode there is data retention), except when it was already in SHUTDOWN state during ACTIVE mode. If it was already in SHUTDOWN during ACTIVE mode, the SRAM remains in SHUTDOWN.	

Table 629. SRAM retention control - Offset: B4h...continued

Bit	Name	Access	Description and value	Reset value
6	RETEN_RAM20	R/W	controls RAM20 power-down modes during low-power modes	0
			0b DEEP SLEEP: The SRAM instance keeps the configuration it has before entering DEEP SLEEP, whatever it is (normal, light sleep, deep-sleep mode and shut down modes). POWER DOWN, DEEP POWER DOWN, and POWEREDOFF: The SRAM instance is in shutdown mode (in this mode there is no data retention).	
			1b The SRAM enters DEEP SLEEP mode during DEEP SLEEP and POWER DOWN state (in this mode there is data retention), except when it was already in SHUTDOWN state during ACTIVE mode. If it was already in SHUTDOWN during ACTIVE mode, the SRAM remains in SHUTDOWN.	
5	RETEN_RAM10	R/W	controls RAM10 power-down modes during low-power modes	0
			0b DEEP SLEEP: The SRAM instance keeps the configuration it has before entering DEEP SLEEP, whatever it is (normal, light sleep, deep-sleep mode and shut down modes). POWER DOWN, DEEP POWER DOWN, and POWEREDOFF: The SRAM instance is in shutdown mode (in this mode there is no data retention).	
			1b The SRAM enters DEEP SLEEP mode during DEEP SLEEP and POWER DOWN state (in this mode there is data retention), except when it was already in SHUTDOWN state during ACTIVE mode. If it was already in SHUTDOWN during ACTIVE mode, the SRAM remains in SHUTDOWN.	

Table 629. SRAM retention control - Offset: B4h...continued

Bit	Name	Access	Description and value	Reset value
4	RETEN_RAM00	R/W	controls RAM00 power-down modes during low-power modes	0
			0b DEEP SLEEP: The SRAM instance keeps the configuration it has before entering DEEP SLEEP, whatever it is (normal, light sleep, deep-sleep mode and shut down modes). POWER DOWN, DEEP POWER DOWN, and POWEREDOFF: The SRAM instance is in shutdown mode (in this mode there is no data retention).	
			1b The SRAM enters DEEP SLEEP mode during DEEP SLEEP and POWER DOWN state (in this mode there is data retention), except when it was already in SHUTDOWN state during ACTIVE mode. If it was already in SHUTDOWN during ACTIVE mode, the SRAM remains in SHUTDOWN.	
3	RETEN_RAMX3	R/W	controls RAMX3 power-down modes during low-power modes	0
			0b DEEP SLEEP: The SRAM instance keeps the configuration it has before entering DEEP SLEEP, whatever it is (normal, light sleep, deep-sleep mode and shut down modes). POWER DOWN, DEEP POWER DOWN, and POWEREDOFF: The SRAM instance is in shutdown mode (in this mode there is no data retention).	
			1b The SRAM enters DEEP SLEEP mode during DEEP SLEEP and POWER DOWN state (in this mode there is data retention), except when it was already in SHUTDOWN state during ACTIVE mode. If it was already in SHUTDOWN during ACTIVE mode, the SRAM remains in SHUTDOWN.	

Table 629. SRAM retention control - Offset: B4h...continued

Bit	Name	Access	Description and value	Reset value
2	RETEN_RAMX2	R/W	controls RAMX2 power-down modes during low-power modes	0
			0b DEEP SLEEP: The SRAM instance keeps the configuration it has before entering DEEP SLEEP, whatever it is (normal, light sleep, deep-sleep mode and shut down modes). POWER DOWN, DEEP POWER DOWN, and POWEREDOFF: The SRAM instance is in shutdown mode (in this mode there is no data retention).	
			1b The SRAM enters DEEP SLEEP mode during DEEP SLEEP and POWER DOWN state (in this mode there is data retention), except when it was already in SHUTDOWN state during ACTIVE mode. If it was already in SHUTDOWN during ACTIVE mode, the SRAM remains in SHUTDOWN.	
1	RETEN_RAMX1	R/W	controls RAMX1 power-down modes during low-power modes	0
			0b DEEP SLEEP: The SRAM instance keeps the configuration it has before entering DEEP SLEEP, whatever it is (normal, light sleep, deep-sleep mode and shut down modes). POWER DOWN, DEEP POWER DOWN, and POWEREDOFF: The SRAM instance is in shutdown mode (in this mode there is no data retention).	
			1b The SRAM enters DEEP SLEEP mode during DEEP SLEEP and POWER DOWN state (in this mode there is data retention), except when it was already in SHUTDOWN state during ACTIVE mode. If it was already in SHUTDOWN during ACTIVE mode, the SRAM remains in SHUTDOWN.	

Table 629. SRAM retention control - Offset: B4h...continued

Bit	Name	Access	Description and value	Reset value
0	RETEN_RAMX0	R/W	controls RAMX0 power-down modes during low-power modes	0
			0b DEEP SLEEP: The SRAM instance keeps the configuration it has before entering DEEP SLEEP, whatever it is (normal, light sleep, deep-sleep mode and shut down modes). POWER DOWN, DEEP POWER DOWN, and POWEREDOFF: The SRAM instance is in shutdown mode (in this mode there is no data retention).	
			1b The SRAM enters DEEP SLEEP mode during DEEP SLEEP and POWER DOWN state (in this mode there is data retention), except when it was already in SHUTDOWN state during ACTIVE mode. If it was already in SHUTDOWN during ACTIVE mode, the SRAM remains in SHUTDOWN.	

45.4.11.2.29 Power-down configuration during active mode (PDRUNCFG0)

Table 630. Power-down configuration during active mode - Offset: B8h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
22	PDEN_RNG	R/W	control power to RNG block	1
			0b RNG module is powered	
			1b RNG module is powered down	
21	PDEN_FLASH	R/W	control power to flash block	0
			0b flash module is powered	
			1b flash module is powered down	
20:18	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
17	PDEN_LDODIG	R/W	control power to LDO_DIG power domain	0
			0b LDO DIG power domain is powered	
			1b LDO DIG power domain is powered down	
16	PDEN_LDOMEM	R/W	control power to LDO MEM power domain	0
			0b LDO MEM power domain is powered	
			1b LDO MEM power domain is powered down	
15:11	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
10	PDEN_BLE	R/W	control power to Bluetooth Low Energy	0
			0b Bluetooth Low Energy retention domain is powered	
			1b Bluetooth Low Energy retention domain is powered down	
9	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
8	PDEN_FRO1M	R/W	Controls power to FRO 1 MHz	0
			0b FRO 1 MHz is powered	
			1b FRO 1 MHz is powered down	
7	PDEN_XTAL32K	R/W	controls power to 32.768 kHz crystal oscillator	1
			0b 32.768 kHz crystal oscillator is powered	
			1b 32.768 kHz crystal oscillator is powered down	
6	PDEN_FRO32K	R/W	controls power to 32.768 kHz FRO	1
			0b 32.768 kHz FRO is powered	
			1b 32.768 KHz FRO is powered down	
5	PDEN_FRO192M	R/W	controls power to 192 MHz free running oscillator	0
			0b FRO 192 MHz is powered	
			1b FRO 192 MHz is powered down	

Table 630. Power-down configuration during active mode - Offset: B8h...continued

Bit	Name	Access	Description and value	Reset value
4	PDEN_BOD2	R/W	controls power to brownout detector 2 (BOD2)	0
			0b BOD2 is powered	
			1b BOD2 is powered down	
3	PDEN_BOD1	R/W	controls power to brownout detector 1 (BOD1)	0
			0b BOD1 is powered	
			1b BOD1 is powered down	
2:1	PDEN_BIAS	R/W	enable or disable bandgap in high-power mode or low-power mode This bit must always be set to 01b (enabled in high accuracy mode) during active state.	1
			00b bias 0.66 V low-power mode enabled Intended to be used for retention mode of memory or digital.	
			01b bias 0.63 V high accuracy mode enabled During active mode, the bias must be set in high accuracy mode.	
			1xb bias disabled	
0	PDEN_DCDC	R/W	Controls power to DCDC converter	0
			0b DCDC is powered	
			1b DCDC is powered down	

45.4.11.2.30 Power-down configuration during active mode set register (PDRUNCFGSET0)

Table 631. Power-down configuration during active mode set register - Offset: C0h

Bit	Name	Access	Description and value	Reset value
31:0	PDRUNCFGSET0	W	Writing ones to this register sets the corresponding bit or bits in the PDRUNCFG0 register, if they are implemented.	0

45.4.11.2.31 Power-down configuration during active mode clear register (PDRUNCFGCLR0)

Table 632. Power-down configuration during active mode clear register - Offset: C8h

Bit	Name	Access	Description and value	Reset value
31:0	PDRUNCFGCLR0	W	Writing ones to this register clears the corresponding bit or bits in the PDRUNCFG0 register, if they are implemented.	0

45.4.11.2.32 ROM control (ROMCTRL)

Table 633. ROM control - Offset: D0h

Bit	Name	Access	Description and value	Reset value
31:7	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
6:3	RM	R/W	read margin control setting	0
2	RME	R/W	read margin enable	0
1	LS	R/W	light sleep (active high) This bit is only used during DfT.	0
0	SDB	R/W	shutdown pin (active low) This bit is only used during DfT.	1

45.4.11.2.33 SRAM control (SRAMCTRL)

Table 634. SRAM control - Offset: D4h

Bit	Name	Access	Description and value	Reset value
31:11	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
10:8	WM	R/W	write margin control settings	7
7:3	RM	R/W	read margin control settings	0xd
2:0	SMB	R/W	source biasing voltage	4

45.4.11.2.34 RAMX0 to RAM20 power modes controls (SRAMCTRL0)

Table 635. RAMX0 to RAM20 power modes controls - Offset: D8h

Bit	Name	Access	Description and value	Reset value
31:28	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
27	RAM20_LSDEL	R/W	RAM20 sleep mode disable	0
26	RAM20_DSBDEL	R/W	RAM20 deep sleep delayed	1
25	RAM20_DSB	R/W	RAM20 deep sleep mode	1
24	RAM20_LS	R/W	RAM20 light sleep mode	0
23	RAM10_LSDEL	R/W	RAM10 sleep mode disable	0
22	RAM10_DSBDEL	R/W	RAM10 deep sleep delayed	1
21	RAM10_DSB	R/W	RAM10 deep sleep mode	1
20	RAM10_LS	R/W	RAM10 light sleep mode	0
19	RAM00_LSDEL	R/W	RAM00 sleep mode disable	0
18	RAM00_DSBDEL	R/W	RAM00 deep sleep delayed	1
17	RAM00_DSB	R/W	RAM00 deep sleep mode	1

Table 635. RAMX0 to RAM20 power modes controls - Offset: D8h...continued

Bit	Name	Access	Description and value	Reset value
16	RAM00_LS	R/W	RAM00 light sleep mode	0
15	RAMX3_LSDEL	R/W	RAMX3 sleep mode disable	0
14	RAMX3_DSBDEL	R/W	RAMX3 deep sleep delayed	1
13	RAMX3_DSB	R/W	RAMX3 deep sleep mode	1
12	RAMX3_LS	R/W	RAMX3 light sleep mode	0
11	RAMX2_LSDEL	R/W	RAMX2 sleep mode disable	0
10	RAMX2_DSBDEL	R/W	RAMX2 deep sleep delayed	1
9	RAMX2_DSB	R/W	RAMX2 deep sleep mode	1
8	RAMX2_LS	R/W	RAMX2 light sleep mode	0
7	RAMX1_LSDEL	R/W	RAMX1 sleep mode disable	0
6	RAMX1_DSBDEL	R/W	RAMX1 deep sleep delayed	1
5	RAMX1_DSB	R/W	RAMX1 deep sleep mode	1
4	RAMX1_LS	R/W	RAMX1 light sleep mode	0
3	RAMX0_LSDEL	R/W	RAMX0 sleep mode disable	0
2	RAMX0_DSBDEL	R/W	RAMX0 deep sleep delayed	1
1	RAMX0_DSB	R/W	RAMX0 deep sleep mode	1
0	RAMX0_LS	R/W	RAMX0 light sleep mode	0

45.4.11.2.35 Bluetooth Low Energy RAM power mode controls (SRAMCTRL1)

Table 636. Bluetooth Low Energy RAM power mode controls - Offset: DCh

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
7	RAMBLESEQL_LSDEL	R/W	RAMBLE sequencer sleep mode disable	0
6	RAMBLESEQL_DSBDEL	R/W	RAM Bluetooth Low Energy sequencer deep sleep delayed	1
5	RAMBLESEQL_DSB	R/W	RAM Bluetooth Low Energy sequencer deep sleep mode	1
4	RAMBLESEQL_LS	R/W	RAM Bluetooth Low Energy sequencer light sleep mode	0
3	RAMBLEDATA_LSDEL	R/W	RAM Bluetooth Low Energy data sleep mode disable	0
2	RAMBLEDATA_DSBDEL	R/W	RAM Bluetooth Low Energy data deep sleep delayed	1
1	RAMBLEDATA_DSB	R/W	RAM Bluetooth Low Energy data deep sleep mode	1

Table 636. Bluetooth Low Energy RAM power mode controls - Offset: DCh...continued

Bit	Name	Access	Description and value	Reset value
0	RAMBLEDATA_LS	R/W	RAM Bluetooth Low Energy data light sleep mode	0

45.4.11.2.36 SRAM assist control 0 (SRAMASSISTCTRL0)

Table 637. SRAM assist control 0 - Offset: E0h

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
5:3	WAM_SRAM	R/W	write assist margin	6
2:1	RAM_SRAM	R/W	read assist margin	0
0	reserved	-	reserved Read value is undefined. Only zeros must be written.	-

45.4.11.2.37 SRAM assist control 1 (SRAMASSISTCTRL1)

Table 638. SRAM assist control 1 - Offset: E4h

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved Read value is undefined. Only zeros must be written.	-
5:3	WAM_SRAM_BLE	R/W	write assist margin	6
2:1	RAM_SRAM_BLE	R/W	read assist margin	0
0	reserved	-	reserved Read value is undefined. Only zeros must be written.	-

45.4.11.2.38 Design for test - enable/disable (DFTENDIS)

Table 639. Design for Test - enable/disable - Offset: F8h

Bit	Name	Access	Description and value	Reset value
31	BLE_OSCTSTEN	R/W	only accessible through DFT	0
30	BLE_EXT_SPI_SEL	R/W	only accessible through DFT	0
29	FRO192M_VSTRESS_CTRL_1	R/W	only accessible through DFT	0
28	FRO192M_VSTRESS_CTRL_0	R/W	only accessible through DFT	0
27	XO32_CAPTEST_VSTRESS_EN	R/W	only accessible through DFT	0
26	FRO32K_VSTRESS_EN	R/W	only accessible through DFT	0
25	FRO1M_VSTRESS_EN	R/W	only accessible through DFT	0
24	DCDC_VSTRESS_EN	R/W	only accessible through DFT	0
23	BOD2_VSTRESS_EN	R/W	only accessible through DFT	0

Table 639. Design for Test - enable/disable - Offset: F8h...continued

Bit	Name	Access	Description and value	Reset value
22	BOD1_VSTRESS_EN	R/W	only accessible through DFT	0
21	BG_STRESS_EN	R/W	only accessible through DFT	0
20	PORLV_VSTRESS_EN	R/W	only accessible through DFT	0
19	PORHV_VSTRESS_EN	R/W	only accessible through DFT	0
18	BLE_STBYEN	R/W	only accessible through DFT	0
17	BLE_DIGCLKEN	R/W	only accessible through DFT	0
16	LP_MODE_FORCE	R/W	only accessible through DFT	0
15	SWCTRL_S13	R/W	only accessible through DFT	0
14	SWCTRL_S12rom	R/W	only accessible through DFT	0
13	SWCTRL_S12	R/W	only accessible through DFT	0
12	SWCTRL_S11	R/W	only accessible through DFT	0
11	SWCTRL_S10	R/W	only accessible through DFT	0
10	SWCTRL_S9	R/W	only accessible through DFT	0
9	SWCTRL_S5b	R/W	only accessible through DFT	0
8	SWCTRL_S5a	R/W	only accessible through DFT	0
7	SWCTRL_S4	R/W	only accessible through DFT	0
6	PD_PMC	R/W	only accessible through DFT	0
5	BUFFER_EN	R/W	only accessible through DFT	0
4	LDOAO_TEST_DIS	R/W	only accessible through DFT	0
3	BG_TEST_EN	R/W	only accessible through DFT	0
2	WUP_TEST_SEL	R/W	only accessible through DFT	0
1	WUP_TEST	R/W	only accessible through DFT	0
0	CORE_OK	R/W	only accessible through DFT	0

45.4.11.2.39 Design for test - DCDC control (DFTDCDC)

Table 640. Design for Test - DCDC control - Offset: FCh

Bit	Name	Access	Description and value	Reset value
31:19	reserved	-	reserved Read value is undefined, only zero must be written.	-
18	BLE_DSLP	R/W	only accessible through DFT	0
17	BG_HP_EN	R/W	only accessible through DFT	0
16	BG_LP_VREF_SW	R/W	only accessible through DFT	0
15	BG_EN	R/W	only accessible through DFT	0
14	LDOM_ENA_LPM	R/W	only accessible through DFT	0
13	LDOM_ENA_HPM	R/W	only accessible through DFT	0

Table 640. Design for Test - DCDC control - Offset: FCh...continued

Bit	Name	Access	Description and value	Reset value
12	LDOD_ENA_LPM	R/W	only accessible through DFT	0
11	LDOD_ENA_HPM	R/W	only accessible through DFT	0
10	BOD_CONNECT2	R/W	only accessible through DFT	0
9	BOD_CONNECT1	R/W	only accessible through DFT	0
8	DCDC_INRUSH_BYP_LV	R/W	only accessible through DFT	0
7	DCDC_SPARE_2	R/W	only accessible through DFT	0
6	DCDC_SPARE_1	R/W	only accessible through DFT	0
5	DCDC_SPARE_0	R/W	only accessible through DFT	0
4	DCDC_TEST_SEL_2	R/W	only accessible through DFT	0
3	DCDC_TEST_SEL_1	R/W	only accessible through DFT	0
2	DCDC_TEST_SEL_0	R/W	only accessible through DFT	0
1	DCDC_TEST_EN	R/W	only accessible through DFT	0
0	DCDC_SW_EN	R/W	only accessible through DFT	0

45.4.11.2.40 Design for test - capacitor test (DFTCAPTEST)

Table 641. Design for test - capacitor test - Offset: 100h

Bit	Name	Access	Description and value	Reset value
31:26	reserved	-	reserved Read value is undefined, only zero must be written.	-
25	COT_VALUE_5	R/W	only accessible through DFT	0
24	COT_VALUE_4	R/W	only accessible through DFT	0
23	COT_VALUE_3	R/W	only accessible through DFT	0
22	COT_VALUE_2	R/W	only accessible through DFT	0
21	COT_VALUE_1	R/W	only accessible through DFT	0
20	COT_VALUE_0	R/W	only accessible through DFT	0
19	reserved	-	reserved Read value is undefined, only zero must be written.	-
18	XO32_CAPTEST_CNTVAL_15	R/W	only accessible through DFT	0
17	XO32_CAPTEST_CNTVAL_14	R/W	only accessible through DFT	0
16	XO32_CAPTEST_CNTVAL_13	R/W	only accessible through DFT	0
15	XO32_CAPTEST_CNTVAL_12	R/W	only accessible through DFT	0
14	XO32_CAPTEST_CNTVAL_11	R/W	only accessible through DFT	0
13	XO32_CAPTEST_CNTVAL_10	R/W	only accessible through DFT	0
12	XO32_CAPTEST_CNTVAL_9	R/W	only accessible through DFT	0

Table 641. Design for test - capacitor test - Offset: 100h...continued

Bit	Name	Access	Description and value	Reset value
11	XO32_CAPTEST_CNTVAL_8	R/W	only accessible through DFT	0
10	XO32_CAPTEST_CNTVAL_7	R/W	only accessible through DFT	0
9	XO32_CAPTEST_CNTVAL_6	R/W	only accessible through DFT	0
8	XO32_CAPTEST_CNTVAL_5	R/W	only accessible through DFT	0
7	XO32_CAPTEST_CNTVAL_4	R/W	only accessible through DFT	0
6	XO32_CAPTEST_CNTVAL_3	R/W	only accessible through DFT	0
5	XO32_CAPTEST_CNTVAL_2	R/W	only accessible through DFT	0
4	XO32_CAPTEST_CNTVAL_1	R/W	only accessible through DFT	0
3	XO32_CAPTEST_CNTVAL_0	R/W	only accessible through DFT	0
2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	XO32_CAPTEST_OSC_IN	R/W	only accessible through DFT	0
0	XO32_CAPTEST_EN	R/W	only accessible through DFT	0

45.4.12 System configuration (SYSCTL)

45.4.12.1 SYSCTL memory map

The SYSCTL registers are located at base address: 4002_3000h

Table 642. System configuration registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
UPDATELOCKOUT	R/W	0h	update lockout control	32		Section 45.4.12.2.1
AUX_FLASH_CTRL	R/W	100h	auxiliary flash control	32		Section 45.4.12.2.2
CODE_GRAY_LSB	R/W	180h	gray code LSB input register	32	0000_0000h	Section 45.4.12.2.3
CODE_GRAY_MSB	R/W	184h	gray code MSB input register	32	0000_0000h	Section 45.4.12.2.4
CODE_BIN_LSB	R	188h	binary code LSB output register	32	0000_0000h	Section 45.4.12.2.5
CODE_BIN_MSB	R	18Ch	binary code MSB output register	32	0000_0000h	Section 45.4.12.2.6
AUX_RNG_CTRL	R/W	200h	auxiliary random number generator control	32	0000_0000h	Section 45.4.12.2.7
AUX_RNG_STATUS	R	208h	auxiliary random number generator status	32	0000_0000h	Section 45.4.12.2.8

45.4.12.2 Register descriptions

45.4.12.2.1 Update lockout control (UPDATELOCKOUT)

Table 643. Update lockout control - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved Read value is undefined, only zero must be written.	-
0	UPDATELOCKOUT	R/W	indicates if all registers in this module are protected	0
			0b normal mode Registers can be written to.	
			1b protected mode Registers cannot be written to.	

45.4.12.2.2 Auxiliary flash control (AUX_FLASH_CTRL)

Table 644. Auxiliary flash control - Offset: 100h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved Read value is undefined, only zero must be written.	-
9	DIS_DATA_SPEC	R/W	disable and clear the flash speculation buffer for data	0
			0b enable buffering data fetches (if speculation is enabled)	
			1b disable buffering data fetches (if speculation is enabled)	
8	DIS_FLASH_SPEC	R/W	disable and clear the flash speculation buffer	0
			0b enable the use of the speculation buffer	
			1b disable the use of the speculation buffer	
7	CLR_FLASH_CACHE	R/W	clear the flash cache	0
			0b no operation	
			1b clear the flash cache	
6	DIS_FLASH_CACHE_DATA	R/W	disable cache for data fetches	0
			0b enable data fetch cache (if cache enabled)	
			1b disable instruction data cache	
5	DIS_FLASH_CACHE_INST	R/W	disable cache for instruction fetches	0
			0b enable instruction fetch cache (if cache enabled)	
			1b disable instruction fetch cache	

Table 644. Auxiliary flash control - Offset: 100h...continued

Bit	Name	Access	Description and value	Reset value
4	DIS_FLASH_CACHE	R/W	disable instruction fetch cache	0
			0b enable flash cache	
			1b disable all flash cache	
3	NPX_CLR_STATE	R/W	clear program model key region	0
			0b no operation	
			1b force clear of all program model key information	
2	DIS_MBECC_BERR_DATA	R/W	bus error on ECC error enable for data fetch	0
			0b Data fetches leading to a flash ECC error terminate in an AHB bus error.	
			1b Data fetches leading to a flash ECC error result in whatever data the flash can return and do not terminate in an AHB bus error.	
1	DIS_MBECC_BERR_INST	R/W	bus error on ECC error enable for instruction fetch	0
			0b Instruction fetches leading to a flash ECC error terminate in an AHB bus error.	
			1b Instruction fetches leading to a flash ECC error result in whatever data the flash can return and do not terminate in an AHB bus error	
0	FLASH_STALL_EN	R/W	flash stall enable	0
			0b flash busy terminates in an AHB bus error	
			1b flash busy stalls until the flash is available	

45.4.12.2.3 Gray code LSB input register (CODE_GRAY_LSB)

Table 645. Gray code LSB input register - Offset: 180h

Bit	Name	Access	Description and value	Reset value
31:0	CODE_GRAY_LSB	R/W	Gray code (42 bits) to be converted back to binary	0

45.4.12.2.4 Gray code MSB input register (CODE_GRAY_MSB)

Table 646. Gray code MSB input register - Offset: 184h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved	-

Table 646. Gray code MSB input register - Offset: 184h...continued

Bit	Name	Access	Description and value	Reset value
9:0	CODE_GRAY_MSB	R/W	Gray code (42 bits) to be converted back to binary	0

45.4.12.2.5 Binary code LSB output register (CODE_BIN_LSB)

Table 647. Binary code LSB output register - Offset: 188h

Bit	Name	Access	Description and value	Reset value
31:0	CODE_BIN_LSB	R	binary converted code (42 bits)	0

45.4.12.2.6 Binary code MSB output register (CODE_BIN_MSB)

Table 648. Binary code MSB output register - Offset: 18Ch

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved	-
9:0	CODE_BIN_MSB	R	binary converted code (42 bits)	0

45.4.12.2.7 Auxiliary random number generator control (AUX_RNG_CTRL)

Table 649. Auxiliary random number generator control - Offset: 200h

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved	-
0	OSC_BYPASS	R/W	oscillator bypass control	1
			0b oscillator bypass is disabled	
			1b oscillator bypass is enabled	

45.4.12.2.8 Auxiliary random number generator status (AUX_RNG_STATUS)

Table 650. Auxiliary random number generator status - Offset: 208h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1	HWERR	R	random number generator signals occurrence of a hardware error	0
			0b no hardware error detected	
			1b hardware error detected	
0	BUSY	R	random number generator is busy	0
			0b random number generator is not busy	
			1b random number generator is busy	

45.4.13 Real-time clock (RTC)

45.4.13.1 RTC memory map

The RTC registers are located at base address: 4002_C000h.

Table 651. Real-time clock registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CTRL	R/W	0h	RTC control register	32		Section 45.4.13.2.1
MATCH	R/W	4h	RTC match register	32	FFFF_FFFFh	Section 45.4.13.2.2
COUNT	R/W	8h	RTC count register	32	0000_0000h	Section 45.4.13.2.3
WAKE	R/W	Ch	high-resolution/wake-up timer control register	32		Section 45.4.13.2.4
SUBSEC	W	10h	subsecond counter register	32		Section 45.4.13.2.5
GPREG0 - GPREG7	R/W	40h - 50h	general-purpose register	32	0000_0000h	Section 45.4.13.2.6

45.4.13.2 Register descriptions

45.4.13.2.1 RTC control register (CTRL)

Table 652. RTC control register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:11	reserved	-	reserved	-
10	RTC_SUBSEC_ENA	R/W	RTC subsecond counter control 0b Subsecond 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). 1b Subsecond counter (if implemented) 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.	0
9	RTC_OSC_BYPASS	R/W	RTC oscillator bypass control 0b The RTC oscillator normally operates as a crystal oscillator with the crystal connected between the RTC_XTALIN and RTC_XTALOUT pins. 1b The RTC oscillator is in bypass mode. In this mode, a clock can be directly input into the RTC_XTALIN pin.	0

Table 652. RTC control register - Offset: 0h...continued

Bit	Name	Access	Description and value		Reset value
8	RTC_OSC_PD	R/W	RTC oscillator power-down control		1
			0b	RTC oscillator is powered up This bit must be cleared in order for the RTC module to function.	
			1b	RTC oscillator is powered-down The RTC oscillator is shut off to reserve power consumption. RTC operation is disabled.	
7	RTC_EN	R/W	RTC enable		0
			0b	disable The RTC 1 Hz and 1 kHz clocks are shut down and the RTC operation is disabled. When writing to load a value in the RTC counter register, this bit must be 0.	
			1b	enable The 1 Hz RTC clock is running 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 enable the high-resolution mode, running from the 1 kHz clock, set CTRL.RTC1KHZ_EN = 1	
6	RTC1KHZ_EN	R/W	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). Note: When running with a 32.768 kHz clock and a divider value of 32, the 1 kHz timer runs at 1.024 kHz.		0
			0b	disable A match on the 1 kHz RTC timer does not bring the part out of deep power-down mode.	
			1b	enable The 1 kHz RTC timer is enabled.	

Table 652. RTC control register - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value
5	WAKEDPD_EN	R/W	RTC 1 kHz timer wake-up enable for deep power-down mode When this bit is set, the RESETCAUSE register is updated. A wake-up from deep power-down is triggered.	0
			0b disable A match on the 1 kHz RTC timer does not bring the part out of deep power-down mode.	
			1b enable A match on the 1 kHz RTC timer brings the part out of deep power-down mode.	
4	ALARMDPD_EN	R/W	RTC 1 Hz timer alarm enable for deep power-down mode	0
			0b disable A match on the 1 Hz RTC timer does not bring the part out of deep power-down mode.	
			1b enable A match on the 1 Hz RTC timer brings the part out of deep power-down mode.	
3	WAKE1KHZ	R/W	RTC 1 kHz timer wake-up flag status	0
			0b run The RTC 1 kHz timer is running. Writing a 0 has no effect.	
			1b timeout 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.	
2	ALARM1HZ	R/W	RTC 1 Hz timer alarm flag status	0
			0b no match No match has occurred on the 1 Hz RTC timer. Writing a 0 has no effect.	
			1b 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 652. RTC control register - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value
1	reserved	-	reserved Read value is undefined, only zero must be written.	-
0	SWRESET	R/W	software reset control not in reset The RTC is not held in reset. This bit must be cleared before configuring or initiating any operation of the RTC. in reset The RTC is held in reset. All register bits within the RTC are forced to their reset value, including writes to set any of the other bits within this register. Do not attempt to write to any bits of this register while the reset bit is cleared.	1

45.4.13.2.2 RTC match register (MATCH)

Table 653. RTC match register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:0	MATVAL	R/W	Contains the value that the 1 Hz RTC timer is compared to. If there is a match, the alarm flag is set (CTRL[RTC_ALARM1HZ] = 1) and generates an alarm interrupt/wake-up request.	1

45.4.13.2.3 RTC counter register (COUNT)

Table 654. RTC counter register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:0	VAL	R/W	A read reflects the current value of the main, 1 Hz RTC timer. A write loads a new initial value into the timer. When the RTC software reset is removed (by clearing bit 0 of the CTRL register), the RTC counter counts up continuously at a 1 Hz rate. 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

45.4.13.2.4 High-resolution/wake-up timer control register (WAKE)

Table 655. High-resolution/wake-up timer control register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15:0	VAL	R/W	A read reflects the current value of the high-resolution/wake-up timer. A write preloads 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

45.4.13.2.5 Subsecond counter register (SUBSEC)

Table 656. Subsecond counter register - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:15	reserved	-	reserved Read value is undefined, only zero must be written.	-
14:0	VAL	R	A read reflects the current value of the 32.768 KHz subsecond counter. Whenever the SUBSEC_ENA bit in the RTC_CONTROL register is low, this counter is cleared. Up-counting at a 32.768 KHz rate commences at the start of the next one-second interval after the SUBSEC_ENA bit is set. This counter must be reenabled after exiting deep power-down mode or after the main RTC module is disabled and reenabled. On modules not equipped with a subsecond counter, this register reads back as all zeroes.	0

45.4.13.2.6 General purpose register (GPREGa)

For a = 0 to 7

Table 657. Subsecond counter register - Offset: 40h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	GPDATA	R/W	data retained during deep power-down mode or loss of main power as long as VBAT is supplied.	0

45.4.14 Synchronous OS/event timer with wake-up timer

45.4.14.1 OSTIMER memory map

The OSTIMER registers are located at base address: 4002_D000h.

Table 658. Synchronous OS/event timer with wake-up timer registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
EVTIMERL	R	0h	central EVTIMER low register	32	0000_0000h	Section 45.4.14.2.1
EVTIMERH	W	4h	central EVTIMER high register	32	0000_0000h	Section 45.4.14.2.2
CAPTURE_L	R	8h	capture low register	32	0000_0000h	Section 45.4.14.2.3
CAPTURE_H	W	Ch	capture high register	32	0000_0000h	Section 45.4.14.2.4
MATCH_L	R/W	10h	match low register	32	FFFF_FFFFh	Section 45.4.14.2.5
MATCH_H	R/W	14h	match high register	32	FFFF_FFFFh	Section 45.4.14.2.6
OSEVENT_CTRL	R/W	1Ch	OS_EVENT TIMER control register	32		Section 45.4.14.2.7

45.4.14.2 Register descriptions

45.4.14.2.1 Central EVTIMER low register (EVTIMERL)

Table 659. EVTIMER low register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:0	EVTIMER_COUNT_VALUE	R	EVTIMER. Note: Only one EVTIMER is readable from all domains.	0

45.4.14.2.2 Central EVTIMER high register (EVTIMERH)

Table 660. EVTIMER high register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved	-
9:0	EVTIMER_COUNT_VALUE	R	A read reflects the current value of the upper 10 bits of the 42-bits EVTIMER. Note: Only one EVTIMER is readable from all domains.	0x0

45.4.14.2.3 Capture low register (CAPTURE_L)

Table 661. Capture low register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:0	CAPTURE_VALUE	R	A read reflects the value of the lower 32 bits of the central 42-bits EVTIMER at the time the last capture signal was generated by the CPU (using CMSIS C function "__SEV();").	0

45.4.14.2.4 Capture high register (CAPTURE_H)

Table 662. Capture high register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved	-
9:0	CAPTURE_VALUE	R	A read reflects the value of the upper 10 bits of the central 42-bits EVTIMER at the time the last capture signal was generated by the CPU (using CMSIS C function "__SEV();").	0

45.4.14.2.5 Match low register (MATCH_L)

Table 663. Match low register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:0	MATCH_VALUE	R/W	The value written to the MATCH (L/H) register pair is compared to the central EVTIMER. When a match occurs, an interrupt request is generated, if enabled.	0

45.4.14.2.6 Match high register (MATCH_H)

Table 664. Match high register - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved	-
9:0	MATCH_VALUE	R/W	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.	1

45.4.14.2.7 OS_EVENT TIMER control register (OSEVENT_CTRL)

Table 665. OS_EVENT TIMER control register - Offset: 1Ch

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved	-
2	MATCH_WR_RDY	R/W	When it is safe to write to reload the match registers, this bit is low. In typical applications it is not necessary to test this bit. The 42-bits 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 must not be initiated until after this transfer completes, which is 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.	0
1	OSTIMER_INTENA	R/W	When this bit is '1', an interrupt/wake-up request to the domain processor is 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.	0
0	OSTIMER_INTRFLAG	R/W	This bit is set when a match occurs between the central 42-bits EVTIMER and the value programmed in the match-register pair. This bit is cleared by writing a '1'. Writes to clear this bit are asynchronous. It must be done before a new match value is written into the MATCH_L/H registers.	0

45.4.15 Flash memory unit (FMU)

45.4.15.1 FMU memory map

The FMU registers are located at base address: 4003_4000h.

Table 666. Flash memory unit registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
FSTAT	W1C	0h	flash status register	32		Section 45.4.15.2.1
FCNFG	R/W	4h	flash configuration register	32		Section 45.4.15.2.2
FCTRL	R/W	8h	flash control register	32		Section 45.4.15.2.3
FTEST	R	Ch	flash test register	32		Section 45.4.15.2.4
FCCOB0	R/W	10h	flash common command object registers	32	0000_0000h	Section 45.4.15.2.5
FCCOB1	R/W	14h	flash common command object registers	32	0000_0000h	Section 45.4.15.2.5
FCCOB2	R/W	18h	flash common command object registers	32	0000_0000h	Section 45.4.15.2.5
FCCOB3	R/W	1Ch	flash common command object registers	32	0000_0000h	Section 45.4.15.2.5
FCCOB4	R/W	20h	flash common command object registers	32	0000_0000h	Section 45.4.15.2.5
FCCOB5	R/W	24h	flash common command object registers	32	0000_0000h	Section 45.4.15.2.5
FCCOB6	R/W	28h	flash common command object registers	32	0000_0000h	Section 45.4.15.2.5
FCCOB7	R/W	2Ch	flash common command object registers	32	0000_0000h	Section 45.4.15.2.5

45.4.15.2 Register descriptions

45.4.15.2.1 Flash status register (FSTAT)

The FSTAT register reports the operational status of the flash module.

Note: When set, the ACCERR, PVIOL, and CMDABT flags prevent the launch of any more commands until the flag is cleared (by writing a 1 to it).

CAUTION: When clearing the DFDIF flag, be careful not to use read-modify-write on the entire FSTAT register and inadvertently clear CCIF to launch a command.

Table 667. Flash status register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31	PERDY	W1C	program-erase ready status flag When a program or sector erase command operation has successfully completed the write phase, the PERDY flag is set by the command controller. The command controller stalls the command operation until the user clears the PERDY flag by writing a 1 to PERDY. Failure to clear PERDY leaves the command controller stalled. However, the flash or IFR space targeted by the command operation remains readable by the FMC. If FCTRL[ABTREQ] is set while the PERDY flag is set, the command controller clears PERDY, aborts the command operation, and sets CCIF and CMDABT.	0
			0b program or sector erase command operation not stalled	
			1b program or sector erase command operation ready to execute	
30:26	reserved	-	reserved	-
25:24	PEWEN	R/W	program-erase write-enable control During program or sector erase command operations, writes are enabled to flash or IFR space in the system memory map. For phrase programming or sector erase, writes are enabled for 4 consecutive words (16 bytes) with address phrase aligned. For page programming, writes are enabled for 8 consecutive phrases (128 bytes) with address page aligned. For programming, data written to erased flash or IFR space for program commands is not available until the program operation has successfully completed. If FMU parameters are not loaded from IFR1 during initialization, PEWEN does not assert.	0
			00b writes are not enabled	
			01b writes are enabled for one flash or IFR phrase (phrase programming, sector erase)	
			10b writes are enabled for one flash or IFR page (page programming)	
			11b reserved	
23:18	reserved	-	reserved	-

Table 667. Flash status register - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value
17	SALV_USED	R	salvage used for erase operation This flag indicates salvage was used during the last erase sector, erase all, or mass erase operation. It is valid at the end of the operation. Salvage involves using ECC to pass erase verify allowing single-bit faults per phrase verified. This flag remains valid until any flash command or mass erase operation launches. Then the flag clears. Using salvage does not guarantee that the erase operation succeeds but only that salvage was used on at least one phrase.	0
			0b salvage not used during last operation	
			1b salvage used during the last erase operation	
16	DFDIF	W1C	double bit fault detect interrupt flag The DFDIF flag indicates an uncorrectable ECC fault was detected during a valid flash read access from the FMC. The DFDIF flag is cleared by writing a 1 to DFDIF. Writing a 0 to DFDIF has no effect.	0
			0b double bit fault not detected during a valid flash read access	
			1b double bit fault detected (or FCTRL[DFDIF] is set) during a valid flash read access	
15:12	CMDDID	R	command domain ID While the CMDP flag is set, the CMDDID bits reflect the domain ID of the write transaction controlling a command write sequence or aborting a command operation. Writes to load an FCCOB register, clear CCIF, clear PERDY, or write to FCTRL[LSACTIVE] must have the same domain ID as the write transaction that started the command write sequence. The CMDDID bits persist after the CMDP flag clears until the CMDP flag sets again at the start of a new command write sequence. Writes to clear the CWSABT, ACCERR, PVIOL, or CMDABT flags are not influenced by the state of the CMDDID bits.	0
11	CMDP	R	command protection status flag The CMDP flag sets when a command write sequence starts with a write to load an existing FCCOB register or clear CCIF while ACCERR, PVIOL, and CMDABT are clear. Clearing the CWSABT, ACCERR, PVIOL, or CMDABT flags as part of a command write sequence do not set the CMDP flag. While the CMDP flag is set, the protection level of any subsequent write transaction to load an FCCOB register, clear CCIF, clear PERDY, or write to FCTRL[LSACTIVE] must match the CMDPRT bits and the domain ID must match the CMDDID bits. The CMDP flag remains set until cleared when CCIF is set after a command operation or power-down recovery completes unless the CMDABT flag is set. The CMDP flag is cleared even if the operation ends with ACCERR or PVIOL set. When the CMDP flag is cleared, the CMDPRT and CMDDID bits remain unchanged. The CMDP flag is also cleared by a mass erase request with CCIF set.	0
			0b command protection level and domain ID are stale	
			1b command protection level (CMDPRT) and domain ID (CMDDID) are set	

Table 667. Flash status register - Offset: 0h...continued

Bit	Name	Access	Description and value		Reset value								
10	reserved	-	reserved		-								
9:8	CMDPRT	R	command protection level While the CMDP flag is set, the CMDPRT bits reflect the secure/privilege protection level of the write transaction controlling a command write sequence or aborting a command operation. Writes to load an existing FCCOB register, clear CCIF, clear PERDY, or write to FCTRL[LSACTIVE] must occur with the same protection level as indicated by the CMDPRT bits. Writes to clear the CWSABT, ACCERR, PVIOL, or CMDABT flags are not influenced by the state of the CMDPRT bits. If FCTRL[ABTREQ] is set while CCIF is high or low, the CMDPRT bits change to reflect the security/privilege protection level of the write transaction that set the ABTREQ bit. <table><tr><td>00b</td><td>secure, normal access</td></tr><tr><td>01b</td><td>secure, privileged access</td></tr><tr><td>10b</td><td>non-secure, normal access</td></tr><tr><td>11b</td><td>non-secure, privileged access</td></tr></table>		00b	secure, normal access	01b	secure, privileged access	10b	non-secure, normal access	11b	non-secure, privileged access	0
00b	secure, normal access												
01b	secure, privileged access												
10b	non-secure, normal access												
11b	non-secure, privileged access												
7	CCIF	W1C	command complete interrupt flag The CCIF flag indicates that a flash command is completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a flash command. The CCIF flag stays low until the command is completed. The CCIF flag is cleared by a mass erase request with CCIF set. The CCIF flag is also cleared upon entry into deep sleep, power-down, and deep power-down modes. It stays low until the end of recovery from these non-active power modes. The CCIF flag is reset to 0 but is set to 1 at the end of flash initialization. <table><tr><td>0b</td><td>flash command, initialization, or power mode recovery in progress</td></tr><tr><td>1b</td><td>Flash command, initialization, or power mode recovery is completed</td></tr></table>		0b	flash command, initialization, or power mode recovery in progress	1b	Flash command, initialization, or power mode recovery is completed	1				
0b	flash command, initialization, or power mode recovery in progress												
1b	Flash command, initialization, or power mode recovery is completed												
6	CWSABT	W1C	command write sequence abort flag The CWSABT flag indicates if a request to abort a command write sequence prior to command launch has been granted. If FCTRL[ABTREQ] is set while CCIF is high, the CWSABT flag is set. When CWSABT is set, ABTREQ is cleared. While the CWSABT flag is set, ABTREQ cannot be set. The CWSABT flag is cleared by writing a 1 to CWSABT, while CCIF is set or cleared and ABTREQ is cleared. Writing a 0 to the CWSABT flag has no effect. The CWSABT flag is cleared by a mass erase request with CCIF set. <table><tr><td>0b</td><td>command write sequence not aborted</td></tr><tr><td>1b</td><td>command write sequence aborted</td></tr></table>		0b	command write sequence not aborted	1b	command write sequence aborted	0				
0b	command write sequence not aborted												
1b	command write sequence aborted												

Table 667. Flash status register - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value
5	ACCERR	W1C	command access error flag The ACCERR flag indicates an illegal attempt was made to launch a flash command due to a violation of the command write sequence or by providing invalid command parameters. The ACCERR flag also sets during command execution if FMU parameters is not loaded from IFR1 during initialization. While the ACCERR flag is set, the CCIF flag cannot be cleared to launch a command. The ACCERR flag is cleared by writing a 1 to ACCERR, while CCIF is set. Writing a 0 to the ACCERR flag has no effect. The ACCERR flag is cleared by a mass erase request with CCIF set. However, it is not cleared during power-down recovery.	0
			0b no access error detected	
			1b access error detected	
4	PVIOL	W1C	command protection violation flag The PVIOL flag indicates an attempt was made to modify a protected area of flash memory during a command operation. While the PVIOL flag is set, the CCIF flag cannot be cleared to launch a command. The PVIOL flag is cleared by writing a 1 to PVIOL, while CCIF is set. Writing a 0 to the PVIOL flag has no effect. The PVIOL flag is cleared by a mass erase request with CCIF set. However, it is not cleared during power-down recovery.	0
3	reserved	-	reserved	-
2	CMDABT	W1C	command abort flag The CMDABT flag indicates FCTRL[ABTREQ] was set during a command operation while CCIF was clear. This event results in the termination of the operation unless it occurs during the exit routine of the operation. When CMDABT is set, ABTREQ is cleared. While the CMDABT flag is set, ABTREQ cannot be set and the CCIF flag cannot be cleared to launch a command. The CMDABT flag is cleared by writing a 1 to CMDABT while CCIF is set and ABTREQ is cleared. Writing a 0 to the CMDABT flag has no effect. The CMDABT flag is cleared by a mass erase request with CCIF set.	0
			0b no command abort detected	
			1b command abort detected	

Table 667. Flash status register - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value				
1	reserved	-	reserved	-				
0	FAIL	R	command fail flag If an error is detected during execution of a flash command, mass erase operation, or flash initialization, the FAIL flag is set . If the FAIL flag is set along with ACCERR after launching a flash command or mass erase operation, FMU parameters have not been loaded from IFR1 during initialization. A clean POR where FMU parameters are successfully loaded is required to clear the FAIL flag. As a status flag, this bit cannot (and is not required to) be cleared by the user like some of the other flags in this register. The value of the FAIL flag for "command-N" is valid only at the end of the "command-N" execution when CCIF = 1 and before the next command has been launched. At some point during the execution of "command-N + 1," the FAIL flag is cleared unless FMU parameters have not been loaded from IFR1 during initialization. The FAIL flag is cleared by a mass erase request with CCIF set, unless FMU parameters have not been loaded from IFR1 during initialization. The FAIL flag is not cleared during power-down recovery. <table><tr><td>0b</td><td>error not detected</td></tr><tr><td>1b</td><td>error detected</td></tr></table>	0b	error not detected	1b	error detected	0
0b	error not detected							
1b	error detected							

45.4.15.2.2 Flash configuration register (FCNFG)

This register provides information on the current functional state of the flash module.

Table 668. Flash configuration register - Offset: 4h

Bit	Name	Access	Description and value		Reset value				
31:28	reserved	-	reserved		-				
27:24	ERSIEN0	R	erase IFR sector enable - block 0 This field controls the ability to erase the IFR sectors in block 0 using the ERSSCR command. These bits are loaded from sideband signals during initialization. Note: The ERSALL command and MCU mass erase request do not adhere to these restrictions. X equals 0, 1, 2, or 3. Any and all bits in the field are configured independently. <table><tr><td>0000b</td><td>block 0 IFR sector X is protected from erase by ERSSCR command</td></tr><tr><td>0001b</td><td>block 0 IFR sector X is not protected from erase by ERSSCR command</td></tr></table>		0000b	block 0 IFR sector X is protected from erase by ERSSCR command	0001b	block 0 IFR sector X is not protected from erase by ERSSCR command	0
0000b	block 0 IFR sector X is protected from erase by ERSSCR command								
0001b	block 0 IFR sector X is not protected from erase by ERSSCR command								
23:17	reserved	-	reserved		-				

Table 668. Flash configuration register - Offset: 4h...continued

Bit	Name	Access	Description and value	Reset value
16	DFDIE	R/W	double bit fault detect interrupt enable When an uncorrectable ECC fault is detected during a valid flash read access from the platform flash controller, the DFDIE bit controls interrupt generation .	0
			0b double bit fault detect interrupt disabled	
			1b double bit fault detect interrupt enabled Whenever the FSTAT[DFDIF] flag is set, an interrupt request is generated.	
15:9	reserved	-	reserved	-
8	ERSREQ	R	mass erase request This bit indicates if a sideband request has been received by the command controller to execute the mass erase operation. ERSREQ is not directly writable but sets when a mass erase request is received by the flash module while CCIF is set. When the operation is complete, ERSREQ is cleared by the command controller.	0
			0b no request or request complete	
			1b request to run the mass erase operation	
7	CCIE	R/W	command complete interrupt enable When a flash command is complete, the CCIE bit controls interrupt generation.	0
			0b command complete interrupt disabled	
			1b command complete interrupt enabled Whenever the FSTAT[CCIF] flag is set, an interrupt request is generated.	
6:0	reserved	-	reserved	-

45.4.15.2.3 Flash control register (FCTRL)

The flash control register controls activity associated with flash memory reads and command operations.

Table 669. Flash control register - Offset: 8h

Bit	Name	Access	Description and value	Reset value				
31:25	reserved	-	reserved	-				
24	ABTREQ	R	abort request With FSTAT[CMDP] set, setting the ABTREQ bit allows a user to abort a command write sequence or command operation. The ABTREQ bit can only be set by a user with the same domain ID as indicated by FSTAT[CMDDID]. It can also only be set by a user with the same or higher indication of FSTAT[CMDPRT] security/privilege protection level. While ABTREQ is set, CCIF cannot be cleared to launch a command. If FSTAT[CCIF] is high when ABTREQ is set, the command write sequence aborts and ownership is transferred to the user setting the ABTREQ bit. The FSTAT[CWSABT] flag is set and the CMDPRT field is updated based on the write transaction that sets the ABTREQ bit. The ABTREQ bit is then cleared automatically. If FSTAT[CCIF] is low when ABTREQ is set, the command operation aborts and ownership is transferred to the user setting the ABTREQ bit. The FSTAT[CMDABT] flag is set and the CMDPRT field is updated based on the write transaction that sets the ABTREQ bit. The ABTREQ bit is then cleared automatically. While CWSABT or CMDABT are high, writes to ABTREQ are ignored. <table><tr><td>0b</td><td>no request to abort a command write sequence</td></tr><tr><td>1b</td><td>request to abort a command write sequence</td></tr></table>	0b	no request to abort a command write sequence	1b	request to abort a command write sequence	1
0b	no request to abort a command write sequence							
1b	request to abort a command write sequence							
23:17	reserved	-	reserved	-				
16	FDFD	R/W	force double bit fault detect The FDFD bit enables the user to emulate the setting of the FSTAT[DFDIF] flag to check the associated interrupt routine. The FDFD bit is cleared by writing a 0 to FDFD. <table><tr><td>0b</td><td>FSTAT[DFDIF] is set only if a double bit fault is detected during a valid flash read access from the platform flash controller.</td></tr><tr><td>1b</td><td>FSTAT[DFDIF] is set during any valid flash read access from the platform flash controller. If the DFDIE bit is set, an interrupt request is generated.</td></tr></table>	0b	FSTAT[DFDIF] is set only if a double bit fault is detected during a valid flash read access from the platform flash controller.	1b	FSTAT[DFDIF] is set during any valid flash read access from the platform flash controller. If the DFDIE bit is set, an interrupt request is generated.	0
0b	FSTAT[DFDIF] is set only if a double bit fault is detected during a valid flash read access from the platform flash controller.							
1b	FSTAT[DFDIF] is set during any valid flash read access from the platform flash controller. If the DFDIE bit is set, an interrupt request is generated.							
15:9	reserved	-	reserved	-				

Table 669. Flash control register - Offset: 8h...continued

Bit	Name	Access	Description and value	Reset value
8	LSACTIVE	R/W	low-speed active mode This bit controls entry into the low-speed active mode. Flash memory reads are supported in low-speed active mode. All flash commands are supported in low-speed active mode. However, the LSACTIVE bit is not writable while a flash command is executing (CCIF = 0) or during a non-active power mode. If the FSTAT[CMDP] is set, the protection level of any write transaction to the LSACTIVE bit must match the CMDPRT bits and the domain ID must match the CMDDDID bits. Note: To guarantee flash operation over the complete voltage and temperature range as defined in the MCX W23 data sheet, this bit must be kept at 1b.	1b
			0b full-speed active mode requested	
			1b low speed active mode requested	
7:4	reserved	-	reserved	-
3:0	RWSC	R/W	read wait-state control These bits control the number of wait-states added to account for the ratio of system clock period to flash access time during full-speed and low-speed active power modes. Ratios greater than one require non-zero settings for the RWSC field for proper flash accesses. The required settings are documented in the device data sheet. The bits below are loaded from IFR1 during initialization.	0
			0h no additional wait-states are added (single cycle access)	
			1h 1 additional wait-state is added	
			2h 2 additional wait-states are added	
			... 15 additional wait-states are added	
			Fh	

45.4.15.2.4 Flash test register (FTEST)

The flash test register is intended to control entry into NVM test mode for the NVM validation team without executing the device sequence to enter functional test mode. The FTEST register is readable if TMECTL is set. TME is writable if TMEWR is set. However, it must only be written while CCIF is set.

Table 670. Flash test register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved	-
4	TMELOCK	R	Test mode entry lock Indicates acceptance of writes to the FTEST register. With TMECTL set, the first write to FTEST must be a 32-bit write of CODE_CODEh. If not, TMELOCK is set and stays set until a reset occurs. If set, writes to FTEST are ignored.	0
			0b FTEST register not locked from accepting writes	
			1b FTEST register locked from accepting writes	
3	TMODE	R	test mode status If TMECTL is set, this bit reflects the state of the ipt_mode_test input.	0
			0b test mode not active	
			1b test mode active	
2	TME	R	test mode entry This bit controls entry into NVM test mode. When TMEWR is set, the TME bit is writable. The TME bit drives the msf_tme_req output. In response, the device must assert the ipt_mode_test input.	0
			0b test mode entry not requested	
			1b test mode entry requested	
1	TMEWR	R	test mode entry writable This bit controls writability of the TME bit. With TMECTL set, TMEWR is set if the first write after reset to FTEST is a 32-bit write of CODE_CODEh. Otherwise, if TMECTL is set, TMELOCK is set and further writes to FTEST are ignored.	0
			0b TME bit is not writable	
			1b TME bit is writable	
0	TMECTL	R	test mode entry control This bit controls readability of the FTEST register and if a 32-bit write of CODE_CODEh enables writability of the TME bit. This bit is loaded from IFR1 during initialization. To prevent the ability to place the flash module in test mode, the IFR1 bit associated with TMECTL must be programmed before shipment.	0
			0h FTEST register always reads 0 and writes to FTEST are ignored	
			1h FTEST register is readable and can be written to enable writability of TME	

45.4.15.2.5 Flash common command object registers (FCCOBa)

The FCCOB register group provides fields for command codes and parameters. The individual words within the set append a 0-7 hex identifier to the FCCOB register name: FCCOB0, FCCOB1, ..., FCCOB7.

For a = 0 to 7

Table 671. Flash common command object registers - Offset: 10h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	CCOBn	R/W	CCOBn The FCCOB register group provides a command code and relevant parameters to the command controller. The individual registers that compose the FCCOB data set can be written in any order. However, all required values, which vary from command to command must be provided. First, set up all required FCCOB fields. Then initiate the command execution by writing a 1 to the FSTAT[CCIF] bit, which clears the CCIF bit, which again locks all FCCOB parameter fields. The fields cannot be changed by the user until the command completes (CCIF returns to 1). No command buffering or queueing is provided. The next command can be loaded only after the current command completes. Some commands return information to the FCCOB registers. Any values returned to FCCOB are available for reading after the FSTAT[CCIF] flag returns to 1. Table 672 shows a generic flash command format. The first FCCOB register, FCCOB0, always contains the command code. This 8-bit value defines the command to be executed. The command code is followed by the parameters required for this specific flash command, typically an address and/or data values.	-

Table 672. Command parameter table

FCCOB number ^[1]	Typical command parameter content [31:0]
0	FCMD [7:0] (flash command code, bits [31:8] are not writable)
1	FCMDOPT [7:0] (flash command options, bits [31:8] are not writable)
2	flash address 1 (start)
3	flash address 2 (end)
4	data word 0
5	data word 1
6	data word 2
7	data word 3

[1] Refers to FCCOB register name, not register address.

45.4.16 NVM PRINCE encryption and decryption (NPX)

45.4.16.1 NPX memory map

The NPX registers are located at base address: 4003_5000h.

Table 673. NVM PRINCE encryption and decryption registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
NPXCR	R/W	0h	NPX control register	32	0000_0000h	Section 45.4.16.2.1
NPXSR	R	4h	NPX status register	32	0000_0004h	Section 45.4.16.2.2
BMAPCTX0	W	10h	bitmap control for memory context 0	32	0000_0000h	Section 45.4.16.2.3
BMAPCTX1	W	14h	bitmap control for memory context 1	32	0000_0000h	Section 45.4.16.2.4
BMAPCTX2	W	18h	bitmap control for memory context 2	32	0000_0000h	Section 45.4.16.2.5
BMAPCTX3	R/W	14h	bitmap control for memory context 3	32	0000_0000h	Section 45.4.16.2.6

45.4.16.2 Register descriptions

45.4.16.2.1 NPX control register (NPXSR)

Contains enables for global lock, system lock, global decryption, and global encryption.

Table 674. NPX control register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:14	reserved	-	reserved This read-only field is reserved and always has the value 0	-
13	GLK	R/W	global lock enable Set by software (sticky). Only cleared by power-on reset (POR).	0
			0b global lock disabled Subsequent reads return 0.	
			1b global lock enabled Cannot write to keys, masks, regions, NPXCR (NPX control register), NPXSR (NPX status register). Subsequent reads return 1.	
12:10	reserved	-	reserved This read-only field is reserved and always has the value 0	-

Table 674. NPX control register - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value
9	SLK	R/W	system lock enable Set by software (sticky). Only cleared by power-on reset (POR).	0
			0b system lock disabled Subsequent reads return 0.	
			1b system lock enabled Cannot write (via IPS) to keys, masks, regions, NPXCR (NPX control register), NPXSR (NPX status register). Subsequent reads return 1.	
8:6	reserved	-	reserved This read-only field is reserved and always has the value 0	-
5	GDE	R/W	global decryption enable Set by software. Cleared by software. Cleared by power-on reset (POR).	0
			0b global decryption disabled NPX on-the-fly decryption is globally disabled. Subsequent reads return 0.	
			1b global decryption enabled NPX on-the-fly decryption is globally enabled. Subsequent reads return 1.	
4:2	reserved	-	reserved This read-only field is reserved and always has the value 0	-
1	GEE	R/W	global encryption enable Set by software. Cleared by software. Cleared by power-on reset (POR).	0
			0b global encryption disabled NPX on-the-fly encryption is globally disabled. Subsequent reads return 0.	
			1b global encryption enabled NPX on-the-fly encryption is globally enabled. Subsequent reads return 1.	
0	reserved	-	reserved This read-only field is reserved and always has the value 0	-

45.4.16.2.2 NPX status register (NPXSR)

Contains Key 0-3 validity flags and the number of supported memory regions (NRGD).

Table 675. NPX status register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:12	reserved	-	reserved This read-only field is reserved and always has the value 0	-
11:8	Vn	R	key n valid Vn is a read-only copy of the corresponding memory context n valid bit of the descriptor.	0
			0b key n is not valid	
			1b key n is valid	
7:4	reserved	-	reserved This read-only field is reserved and always has the value 0	-
3:1	NUMCTX	R	Number of implemented memory contexts Contains the number of supported memory contexts. In this device, there are 4 memory contexts.	0
			0000b no (zero) implemented memory contexts	
			0001b 1 implemented memory context	
			0010b 2 implemented memory contexts	
			0011b 3 implemented memory contexts	
			0100b 4 implemented memory contexts	
0	reserved	-	reserved This read-only field is reserved and always has the value 0	-

45.4.16.2.3 Bitmap control for memory context 0 (BMAPCTX0)

Contains a 32-bit bitmap that controls if the corresponding 8-KByte block of flash (out of a maximum size of 256 KBytes) is encrypted/decrypted within memory context 0 using KEY0.

Table 676. Bitmap control for memory context 0 - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:0	BENn	W	block enable for encryption/decryption Set bit n to enable encryption/decryption for the corresponding 8-KByte block n.	0
			0b disable encryption/decryption	
			1b enable encryption/decryption	

45.4.16.2.4 Bitmap control for memory context 1 (BMAPCTX1)

Contains a 32-bit bitmap that controls if the corresponding 8-KByte block of flash (out of a maximum size of 256 KBytes) is encrypted/decrypted within memory context 1 using KEY1.

Table 677. Bitmap control for memory context 1 - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:0	BENn	W	block enable for encryption/decryption Set bit n to enable encryption/decryption for the corresponding 8-KByte block n.	0
			0b disable encryption/decryption	
			1b enable encryption/decryption	

45.4.16.2.5 Bitmap control for memory context 2 (BMAPCTX2)

Contains a 32-bit bitmap that controls if the corresponding 8-KByte block of flash (out of a maximum size of 256 KBytes) is encrypted/decrypted within memory context 2 using KEY2.

Table 678. Bitmap control for memory context 2 - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:0	BENn	W	block enable for encryption/decryption Set bit n to enable encryption/decryption for the corresponding 8-KByte block n.	0
			0b disable encryption/decryption	
			1b enable encryption/decryption	

45.4.16.2.6 Bitmap control for memory context 3 (BMAPCTX3)

Contains a 32-bit bitmap that controls if the corresponding 8-KByte block of flash (out of a maximum size of 256 KBytes) is encrypted/decrypted within memory context 3 using KEY3.

Table 679. Bitmap control for memory context 3 - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:0	BENn	W	block enable for encryption/decryption Set bit n to enable encryption/decryption for the corresponding 8-KByte block n.	0
			0b disable encryption/decryption	
			1b enable encryption/decryption	

45.4.17 True random number generator (TRNG)

45.4.17.1 TRNG memory map

The TRNG registers are located at base address: 4003_A000h.

Table 680. True random number generator registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
MCTL	R/W	0h	miscellaneous control register	32	0001_2001h	Section 45.4.17.2.1
SCMISC	R/W	4h	statistical check miscellaneous register	32	0001_0022h	Section 45.4.17.2.2
PKRRNG	R/W	8h	poker range register	32	0000_09A3h	Section 45.4.17.2.3
PKRMAX	R/W	Ch	poker maximum limit register	32	0000_6920h	Section 45.4.17.2.4
PKRSQ	R	Ch	poker square calculation result register	32	0000_0000h	Section 45.4.17.2.5
SDCTL	R/W	10h	seed control register	32	0C80_09C4h	Section 45.4.17.2.6
SBLIM	R/W	14h	sparse bit limit register	32	0000_003Fh	Section 45.4.17.2.7
TOTSAM	R	14h	total samples register	32	0000_0000h	Section 45.4.17.2.8
FRQMIN	R/W	18h	frequency count minimum limit register	32	0000_0640h	Section 45.4.17.2.9
FRQCNT	R	1Ch	frequency count register	32	0000_0000h	Section 45.4.17.2.10
FRQMAX	R/W	1Ch	frequency count maximum limit register	32	0000_6400h	Section 45.4.17.2.11
SCMC	R	20h	statistical check monobit count register	32	0000_0000h	Section 45.4.17.2.12
SCML	R/W	20h	statistical check monobit limit register	32	010C_0568h	Section 45.4.17.2.13
SCR1C	R	24h	statistical check run length 1 count register	32	0000_0000h	Section 45.4.17.2.14
SCR1L	R/W	24h	statistical check run length 1 limit register	32	00B2_0195h	Section 45.4.17.2.15
SCR2C	R	28h	statistical check run length 2 count register	32	0000_0000h	Section 45.4.17.2.16
SCR2L	R/W	28h	statistical check run length 2 limit register	32	007A_00DCh	Section 45.4.17.2.17
SCR3C	R	2Ch	statistical check run length 3 count register	32	0000_0000h	Section 45.4.17.2.18
SCR3L	R/W	2Ch	statistical check run length 3 limit register	32	0058_007Dh	Section 45.4.17.2.19
SCR4C	R	30h	statistical check run length 4 count register	32	0000_0000h	Section 45.4.17.2.20
SCR4L	R/W	30h	statistical check run length 4 limit register	32	0040_004Bh	Section 45.4.17.2.21

Table 680. True random number generator registers...continued

Name	Access	Offset	Description	Width (bits)	Reset value	Section
SCR5C	R	34h	statistical check run length 5 count register	32	0000_0000h	Section 45.4.17.2.22
SCR5L	R/W	34h	statistical check run length 5 limit register	32	002E_002Fh	Section 45.4.17.2.23
SCR6PC	R	38h	statistical check run length 6+ count register	32	0000_0000h	Section 45.4.17.2.24
SCR6PL	R/W	38h	statistical check run length 6+ limit register	32	002E_002Fh	Section 45.4.17.2.25
STATUS	R	3Ch	status register	32	0000_0000h	Section 45.4.17.2.26
ENT0 - ENT15	R	40h - 7Ch	entropy read register	32	0000_0000h	Section 45.4.17.2.27
PKRCNT10	R	80h	statistical check poker count 1 and 0 register	32	0000_0000h	Section 45.4.17.2.28
PKRCNT32	R	84h	statistical check poker count 3 and 2 register	32	0000_0000h	Section 45.4.17.2.29
PKRCNT54	R	88h	statistical check poker count 5 and 4 register	32	0000_0000h	Section 45.4.17.2.30
PKRCNT76	R	8Ch	statistical check poker count 7 and 6 register	32	0000_0000h	Section 45.4.17.2.31
PKRCNT98	R	90h	statistical check poker count 9 and 8 register	32	0000_0000h	Section 45.4.17.2.32
PKRCNTBA	R	94h	statistical check poker count B and A register	32	0000_0000h	Section 45.4.17.2.33
PKRCNTDC	R	98h	statistical check poker count D and C register	32	0000_0000h	Section 45.4.17.2.34
PKRCNTFE	R	9Ch	statistical check poker count F and E register	32	0000_0000h	Section 45.4.17.2.35
SEC_CFG	R/W	A0h	security configuration register	32	0000_0000h	Section 45.4.17.2.36
INT_CTRL	R/W	A4h	interrupt control register	32	0000_0007h	Section 45.4.17.2.37
INT_MASK	R/W	A8h	mask register	32	0000_0000h	Section 45.4.17.2.38
INT_STATUS	R	ACh	interrupt status register	32	0000_0000h	Section 45.4.17.2.39
VID1	R	F0h	version ID register (MS)	32	0030_0301h	Section 45.4.17.2.40
VID2	R	F4h	version ID register (LS)	32	0000_0000h	Section 45.4.17.2.41

45.4.17.2 Register descriptions

45.4.17.2.1 Miscellaneous control register (MCTL)

This register is intended to be used for programming, configuring, and testing the RNG. It is the main register to read/write to enable entropy generation, to stop entropy generation, and to block access to entropy registers. These actions are done via the special TRNG_ACC and PRGM bit below.

The miscellaneous control register is a read/write register used to control the true random number generator (TRNG) access, operation, and testing of the RNG.

Note: In many cases two RNG registers share the same address. A particular register at the shared address is selected based upon the value in the PRGM field of the MCTL register.

Table 681. Miscellaneous control register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:17	reserved	-	reserved	-
16	PRGM	R/W	programming mode select When this bit is 1, the TRNG is in program mode. Otherwise, it is in run mode. No entropy value is generated while the TRNG is in program mode. Note: Different RNG registers are accessible at the same address depending on if PRGM is set to 1 or 0. This feature is noted in the RNG register descriptions.	1
15	reserved	-	reserved	-
14	LRUN_CONT	R/W	long run count continues between entropy generations If set to 1, long_run count of the internal test at the start of an entropy generation cycle continues where it left off at the end of the previous entropy generation. If set to 0, normal LRUN count is restarted for each entropy regeneration.	0
13	TSTOP_OK	R	TRNG_OK_TO_STOP Software can check that this bit is a 1 before transitioning TRNG to low-power mode (TRNG clock stopped). TRNG turns on the TRNG free-running ring oscillator whenever a new entropy is generated and turns off the ring oscillator at the completion of the entropy generation is complete. If the TRNG clock is stopped while the TRNG ring oscillator is running, the oscillator continues running even though the TRNG clock is stopped. To ensure that the ring oscillator is stopped, assert ipg_stop input and verify that ipg_rng_stop_ack is asserted after at least two (2) clock cycles. When the TRNG ring oscillator is not running, TSTOP_OK is asserted. It helps for cases where the TRNG clock must be stopped without having access to ipg_stop nor ipg_rng_stop_ack.	1
12	ERR	W1C	error detection read: error status 1 = error detected. 0 = no error. write Write 1 to clear errors. Writing 0 has no effect.	0
11	TST_OUT	R	test point inside ring oscillator	0

Table 681. Miscellaneous control register - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value
10	ENT_VAL	R	entropy valid Asserts only if the TRNG ACC bit is set and then after an entropy value is generated. Is cleared at maximum one (1) bus clock cycle after reading the ENT15 register (ENT0 through ENT14 must be read before reading ENT15).	0
9	FCT_VAL	R	frequency count valid Indicates that a valid frequency count may be read from FRQCNT.	0
8	FCT_FAIL	R	frequency count fail	0
7	FOR_SCLK	R/W	force system clock If set, the system clock is used to operate the TRNG, instead of the ring oscillator. It is for test use only. Indeterminate results may occur. This bit is writable only if PRGM bit is 1 or the PRGM bit is written to 1 simultaneously to writing this bit. This bit is cleared by writing the RST_DEF bit to 1.	0
6	RST_DEF	W	reset defaults Writing a 1 to this bit clears various TRNG registers and bits within the registers to their default state. This bit is writable only if PRGM bit is 1 or PRGM bit is written to 1 simultaneously to writing this bit. Reading this bit always produces a 0.	0
5	TRNG_ACC	R/W	TRNG access mode If this bit is set to 1, the TRNG generates an entropy value that can be read via the ENT0-ENT15 registers. The entropy value may be read once the ENT VAL bit is asserted. Also see ENTa register descriptions (For a = 0 to 15).	0
4	UNUSED4	R	This bit is unused. Always reads zero.	0
3:2	OSC_DIV	R/W	oscillator divide Determines the dividing done to the ring oscillator before it is used by the TRNG. This field is writable only if PRGM bit is 1 or PRGM bit is written to 1 simultaneously to writing this field. This field is cleared to the default POR value by writing the RST_DEF bit to 1.	0
			00b use ring oscillator with no divide	
			01b use ring oscillator divided-by-2	
			10b use ring oscillator divided-by-4	
			11b use ring oscillator divided-by-8	

Table 681. Miscellaneous control register - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value								
1:0	SAMP_MODE	R/W	sample mode Determines the method of sampling the ring oscillator while generating the entropy value. This field is writable only if PRGM bit is 1 or PRGM bit is written to 1 simultaneously to writing this field. This field is cleared to the default POR value by writing the RST_DEF bit to 1. <table><tr><td>00b</td><td>use Von Neumann data into entropy shifter and statistical checker</td></tr><tr><td>01b</td><td>use raw data into entropy shifter and statistical checker</td></tr><tr><td>10b</td><td>use Von Neumann data into entropy shifter; use raw data into statistical checker</td></tr><tr><td>11b</td><td>undefined/reserved</td></tr></table>	00b	use Von Neumann data into entropy shifter and statistical checker	01b	use raw data into entropy shifter and statistical checker	10b	use Von Neumann data into entropy shifter; use raw data into statistical checker	11b	undefined/reserved	0
00b	use Von Neumann data into entropy shifter and statistical checker											
01b	use raw data into entropy shifter and statistical checker											
10b	use Von Neumann data into entropy shifter; use raw data into statistical checker											
11b	undefined/reserved											

45.4.17.2.2 Statistical check miscellaneous register (SCMISC)

The statistical check miscellaneous register contains the long run maximum limit value and the retry count value. This register is only accessible when the MCTL[PRGM] bit is 1. Otherwise this register reads zeroes and cannot be written.

Table 682. Statistical check miscellaneous register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:20	reserved	-	reserved	-
19:16	RTY_CT	R/W	retry count If a statistical check fails during the TRNG entropy generation, the RTY_CT value indicates the number of times a retry occurs before an error is generated. This field is only writable when MCTL[PRGM] = 1. This field reads zeroes when MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0
15:8	reserved	-	reserved	-
7:0	LRUN_MAX	R/W	long run maximum limit This value is the largest allowable number of consecutive samples of all 1, or all 0, that is allowed during the entropy generation. This field is only writable when MCTL[PRGM] = 1. This field reads zeroes when MCTL[PRGM] = 0. This field is cleared to the POR reset value by writing the MCTL[RST_DEF] bit to 1.	0

45.4.17.2.3 Poker range register (PKRRNG)

The poker range register defines the difference between the TRNG poker maximum limit and the minimum limit. These limits are used during the TRNG statistical check poker test.

Table 683. Poker range register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved; always 0	-
15:0	PKR_RNG	R/W	poker range During the TRNG statistical checks, a "poker test" is run which requires a maximum and minimum limit. The maximum is programmed in the PKRMAX[PKR_MAX] register. The minimum is derived by subtracting the PKR_RNG value from the programmed maximum value. This field is only writable when MCTL[PRGM] = 1. This field reads zeroes when MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note: The minimum allowable poker result is $PKR\_MAX - PKR\_RNG + 1$.	0

45.4.17.2.4 Poker maximum limit register (PKRMAX)

The poker maximum limit register defines maximum limit allowable during the TRNG statistical check poker test. Note that this offset (0x0C) is used as PKRMAX only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as the PKRSQ readback register.

Note: This offset (0x0C) is only used as PKRMAX when MCTL[PRGM] = 1. When MCTL[PRGM] = 0, this offset is used as the PKRSQ readback register (see [Section 45.4.17.2.5](#)).

Table 684. Poker maximum limit register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:24	reserved	-	reserved; always 0	-
23:0	PKR_MAX	R/W	poker maximum limit During the TRNG statistical checks, a "poker test" is run which requires a maximum and minimum limit. The maximum allowable result is programmed in the PKRMAX[PKR_MAX] register. This field is only writable when MCTL[PRGM] = 1. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note: The PKRMAX and PKRRNG registers combined are used to define the minimum allowable poker result, which is $PKR\_MAX - PKR\_RNG + 1$. Note: If MCTL[PRGM] bit is 0, this register address is used to read the poker test square calculation result in register PKRSQ, as defined in Section 45.4.17.2.5.	0

45.4.17.2.5 Poker square calculation result register (PKRSQ)

The Poker Square Calculation Result Register is a read-only register used to read the result of the TRNG Statistical Check Poker Test's Square Calculation. This test starts with the PKRMAX value and decreases towards a final result, which is read here. For the Poker Test to pass, this final result must be less than the programmed PKRRNG value.

Note: This offset (0x0C) is used as PKRMAX if MCTL[PRGM] = 1 (see [Section 45.4.17.2.4](#)). If MCTL[PRGM] = 0, this offset is used as PKRSQ readback register.

Table 685. Poker square calculation result register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:24	reserved	-	reserved; always 0	-
23:0	PKR_SQ	R	poker square calculation result During the TRNG statistical checks, a "poker test" is run which starts with the value PKRMAX[PKR_MAX]. This value decreases according to a "sum of squares" algorithm. It must remain greater than zero, but less than the PKRRNG[PKR_RNG] limit. When MCTL[PRGM] = 0, the resulting value may be read through this register. Note: if MCTL[PRGM] = 1, this register address is used to access the poker test maximum limit in register PKRMAX, as defined in Section 45.4.17.2.4.	0

45.4.17.2.6 Seed control register (SDCTL)

The seed control register contains two fields. One field defines the length (in system clocks) of each entropy sample (ENT_DLY). The other field indicates the number of samples that is taken during each TRNG entropy generation (SAMP_SIZE).

Table 686. Seed control register - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:16	ENT_DLY	R/W	entropy delay Defines the length (in system clocks) of each Entropy sample taken. This field is only writable when MCTL[PRGM] = 1. This field reads zeroes when MCTL[PRGM] = 0. This field is cleared to its reset value at POR.	0
23:0	SAMP_SIZE	R/W	sample size Defines the total number of entropy samples that is taken during entropy generation. This field is only writable when MCTL[PRGM] = 1. This field reads zeroes when MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0

45.4.17.2.7 Sparse bit limit register (SBLIM)

The sparse bit limit register is used when Von Neumann sampling is selected during entropy generation. It defines the maximum number of consecutive Von Neumann samples which may be discarded before an error is generated.

Note: This address (0x14) is only used as SBLIM when MCTL[PRGM] = 1. When MCTL[PRGM] = 0, this address is used as TOTSAM readback register (see [Section 45.4.17.2.8](#)).

Table 687. Sparse bit limit register - Offset: 14h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved; always 0	-
9:0	SB_LIM	R/W	sparse bit limit During Von Neumann sampling (if enabled by MCTL[SAMP_MODE]), samples are discarded if two consecutive raw samples are both 0 or both 1. If this discarding occurs for a long period of time, it indicates that there is insufficient entropy. The sparse bit limit defines the maximum number of consecutive samples that may be discarded before an error is generated. This field is only writable when MCTL[PRGM] = 1. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note: When MCTL[PRGM] = 0, this register address is used to read the total samples count in register TOTSAM, as defined in Section 45.4.17.2.8.	0

45.4.17.2.8 Total samples register (TOTSAM)

The total samples register is a read-only register used to read the total number of samples taken during entropy generation. It is used to give an indication of how often a sample is actually used during Von Neumann sampling.

Note: This offset (0x14) is used as SBLIM when MCTL[PRGM] = 1 (see [Section 45.4.17.2.7](#)). When MCTL[PRGM] = 0, this offset is used as TOTSAM readback register.

Table 688. Total samples register - Offset: 14h

Bit	Name	Access	Description and value	Reset value
31:20	reserved	-	reserved; always 0	-
19:0	TOT_SAM	R	total samples During entropy generation, the total number of raw samples is counted. This count is useful for determining how often a sample is used during Von Neumann sampling. When MCTL[PRGM] = 0, the count may be read through this register. Note: When MCTL[PRGM] = 1, this register address is used to access the sparse bit limit in register SBLIM, as defined in Section 45.4.17.2.7.	0

45.4.17.2.9 Frequency count minimum limit register (FRQMIN)

The frequency count minimum limit register defines the minimum allowable count taken by the entropy sample counter during each entropy sample. During any sample period, if the count is less than this programmed minimum, a frequency count fail is flagged in MCTL[FCT_FAIL] and an error is generated.

Table 689. Frequency count minimum limit register - Offset: 18h

Bit	Name	Access	Description and value	Reset value
31:22	reserved	-	reserved; always 0	-
21:0	FRQ_MIN	R/W	frequency count minimum limit Defines the minimum allowable count taken during each entropy sample. This field is only writable when MCTL[PRGM] = 1. This field reads zeroes when MCTL[PRGM] = 0. This field is cleared to its reset value at POR.	0

45.4.17.2.10 Frequency count register (FRQCNT)

The frequency count register is a read-only register used to read the frequency counter within the TRNG entropy generator. It reads all zeroes unless PRGM[MCTL] = 0 and MCTL[TRNG_ACC] = 1. This register must be read once, as a full 32-bit. A subsequent read is only valid when MCTL[FCT_VAL] bit is HIGH. A read clears this register. The next read when MCTL[FCT_VAL] bit is HIGH must be treated as a different value from the previous one, even though the counts might be the same.

Note: This offset (0x1C) is used as FRQMAX when MCTL[PRGM] = 1 (see [Section 45.4.17.2.11](#)). When MCTL[PRGM] = 0, this offset is used as live FRQCNT readback register.

Table 690. Frequency count register - Offset: 1Ch

Bit	Name	Access	Description and value	Reset value
31:22	reserved	-	reserved; always 0	-
21:0	FRQ_CT	R	frequency count When MCTL[TRNG_ACC] = 1, reads a sample frequency count taken during entropy generation. Requires MCTL[PRGM] = 0. Note: When MCTL[PRGM] = 1, this register address is used to access the frequency maximum limit in register FRQMAX, as defined in the FRQ_MAX field .	0

45.4.17.2.11 Frequency count maximum limit register (FRQMAX)

The frequency count maximum limit register defines the maximum allowable count taken by the entropy sample counter during each entropy sample. During any sample period, if the count is greater than this programmed maximum, a frequency count fail is flagged in MCTL[FCT_FAIL] and an error is generated.

Note: This address (001C) is only used as FRQMAX when MCTL[PRGM] = 1. When MCTL[PRGM] = 0, this address is used as FRQCNT readback register (see [Section 45.4.17.2.10](#)).

Table 691. Frequency count maximum limit register - Offset: 1Ch

Bit	Name	Access	Description and value	Reset value
31:22	reserved	-	reserved; always 0	-
21:0	FRQ_MAX	R/W	frequency counter maximum limit Defines the maximum allowable count taken during each entropy sample. This field is writable only if MCTL[PRGM] bit is 1. This field is cleared to its reset value at POR. Note: When MCTL[PRGM] = 0, this register address is used to read the frequency count result in register FRQCNT, as defined in Section 45.4.17.2.10 .	0

45.4.17.2.12 Statistical check monobit count register (SCMC)

The statistical check monobit count register is a read-only register used to read the final monobit count after entropy generation. This counter starts with the value in SCML[MONO_MAX]. It is decremented each time a 1 is sampled.

Note: This offset (0x20) is used as SCML when MCTL[PRGM] = 1 (see [Section 45.4.17.2.13](#)). When MCTL[PRGM] = 0, this offset is used as SCMC readback register.

Table 692. Statistical check monobit count register - Offset: 20h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved; always 0	-
15:0	MONO_CT	R	monobit count Reads the final monobit count after entropy generation. Requires MCTL[PRGM] = 0. Note: When MCTL[PRGM] = 1, this register address is used to access the statistical check monobit limit in register SCML, as defined in Section 45.4.17.2.13 .	0

45.4.17.2.13 Statistical check monobit limit register (SCML)

The statistical check monobit limit register defines the allowable maximum and minimum number of 1s/0s detected during entropy generation. To pass the test, the number of 1s/0s generated must be less than the programmed maximum value. The number of 1s/0s generated must be greater than (maximum - range). If this test fails, the retry counter in SCMISC is decremented. If the retry count has not reached zero, a retry occurs. If the retry count has reached zero, an error is generated.

Note: This offset (0x20) is only used as SCML when MCTL[PRGM] = 1. When MCTL[PRGM] = 0, this offset is used as SCMC readback register (see [Section 45.4.17.2.2](#)).

Table 693. Statistical check monobit limit register - Offset: 20h

Bit	Name	Access	Description and value	Reset value
31:16	MONO_RNG	R/W	monobit range The number of 1s/0s detected during entropy generation must be greater than MONO_MAX – MONO_RNG. If not, a retry or error occurs. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0
15:0	MONO_MAX	R/W	monobit maximum limit Defines the maximum allowable count taken during entropy generation. The number of 1s/0s detected during entropy generation must be less than MONO_MAX. If not, a retry or error occurs. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0

45.4.17.2.14 Statistical check run length 1 count register (SCR1C)

The statistical check run length 1 counters register is a read-only register used to read the final run length 1 counts after entropy generation. These counters start with the value in SCR1L[RUN1_MAX]. The R1_1_CT decrements each time a single one is sampled (preceded by a zero and followed by a zero). The R1_0_CT decrements each time a single zero is sampled (preceded by a one and followed by a one).

Note: This offset (0x24) is used as SCR1L when MCTL[PRGM] = 1 (see [Section 45.4.17.2.15](#)). When MCTL[PRGM] = 0, this offset is used as SCR1C readback register.

Table 694. Statistical check run length 1 count register - Offset: 24h

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved; always 0	-
30:16	R1_1_CT	R	runs of 1, length 1 count Reads the final runs of 1s, length 1 count after entropy generation. Requires MCTL[PRGM] = 0.	0
15	reserved	-	reserved; always 0	-
14:0	R1_0_CT	R	runs of 0, length 1 count Reads the final runs of 0s, length 1 count after entropy generation. Requires MCTL[PRGM] = 0.	0

45.4.17.2.15 Statistical check run length 1 limit register (SCR1L)

The statistical check run length 1 limit register defines the allowable maximum and minimum number of runs of length 1 detected during entropy generation. To pass the test, the number of runs of length 1 (for samples of both 0 and 1) must be less than the programmed maximum value. The number of runs of length 1 must be greater than (maximum – range). If this test fails, the retry counter in SCMISC is decremented. If the retry count has not reached zero, a retry occurs. If the retry count has reached zero, an error is generated.

Note: This address (0x24) is only used as SCR1L when MCTL[PRGM] = 1. When MCTL[PRGM] = 0, this address is used as SCR1C readback (see [Section 45.4.17.2.14](#)) register.

Table 695. Statistical check run length 1 limit register - Offset: 24h

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved; always 0	-
30:16	RUN1_RNG	R/W	run length 1 range The number of runs of length 1 (for both 0 and 1) detected during entropy generation must be greater than RUN1_MAX – RUN1_RNG. If not, a retry or error occurs. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0
15	reserved	-	reserved; always 0	-
14:0	RUN1_MAX	R/W	run length 1 maximum limit Defines the maximum allowable runs of length 1 (for 0 and 1) detected during entropy generation. The number of runs of length 1 detected during entropy generation must be less than RUN1_MAX. If not, a retry or error occurs. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0

45.4.17.2.16 Statistical check run length 2 count register (SCR2C)

The statistical check run length 2 counters register is a read-only register used to read the final run length 2 counts after entropy generation. These counters start with the value in SCR2L[RUN2_MAX]. The R2_1_CT decrements each time a single one is sampled (preceded by a zero and followed by a zero). The R2_0_CT decrements each time a single zero is sampled (preceded by a one and followed by a one).

Note: This offset (0x24) is used as SCR2L when MCTL[PRGM] = 1 (see [Section 45.4.17.2.17](#)). When MCTL[PRGM] = 0, this offset is used as SCR2C readback register.

Table 696. Statistical check run length 2 count register - Offset: 28h

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved; always 0	-
30:16	R2_1_CT	R	runs of 1, length 2 count Reads the final runs of 1s, length 2 count after entropy generation. Requires MCTL[PRGM] = 0.	0
15	reserved	-	reserved; always 0	-
14:0	R2_0_CT	R	runs of 0, length 2 count Reads the final runs of 0s, length 2 count after entropy generation. Requires MCTL[PRGM] = 0.	0

45.4.17.2.17 Statistical check run length 2 limit register (SCR2L)

The statistical check run length 2 limit register defines the allowable maximum and minimum number of runs of length 2 detected during entropy generation. To pass the test, the number of runs of length 2 (for samples of both 0 and 1) must be less than the programmed maximum value. The number of runs of length 2 must be greater than (maximum – range). If this test fails, the retry counter in SCMISC is decremented. If the retry count has not reached zero, a retry occurs. If the retry count has reached zero, an error is generated.

Note: This address (0x24) is only used as SCR2L when MCTL[PRGM] = 1. When MCTL[PRGM] = 0, this address is used as SCR2C readback (see [Section 45.4.17.2.16](#)) register.

Table 697. Statistical check run length 2 limit register - Offset: 28h

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved; always 0	-
30:16	RUN2_RNG	R/W	run length 2 range The number of runs of length 2 (for both 0 and 1) detected during entropy generation must be greater than RUN2_MAX – RUN2_RNG. If not, a retry or error occurs. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0
15	reserved	-	reserved; always 0	-
14:0	RUN2_MAX	R/W	run length 2 maximum limit Defines the maximum allowable runs of length 2 (for 0 and 1) detected during entropy generation. The number of runs of length 2 detected during entropy generation must be less than RUN2_MAX. If not, a retry or error occurs. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0

45.4.17.2.18 Statistical check run length 3 count register (SCR3C)

The statistical check run length 3 counters register is a read-only register used to read the final run length 3 counts after entropy generation. These counters start with the value in SCR3L[RUN3_MAX]. The R3_1_CT decrements each time a single one is sampled (preceded by a zero and followed by a zero). The R3_0_CT decrements each time a single zero is sampled (preceded by a one and followed by a one).

Note: This offset (0x24) is used as SCR3L when MCTL[PRGM] = 1 (see [Section 45.4.17.2.19](#)). When MCTL[PRGM] = 0, this offset is used as SCR3C readback register.

Table 698. Statistical check run length 3 count register - Offset: 2Ch

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved; always 0	-
30:16	R3_1_CT	R	runs of 1, length 3 count Reads the final runs of 1s, length 3 count after entropy generation. Requires MCTL[PRGM] = 0.	0
15	reserved	-	reserved; always 0	-
14:0	R3_0_CT	R	runs of 0, length 3 count Reads the final runs of 0s, length 3 count after entropy generation. Requires MCTL[PRGM] = 0.	0

45.4.17.2.19 Statistical check run length 3 limit register (SCR3L)

The statistical check run length 3 limit register defines the allowable maximum and minimum number of runs of length 3 detected during entropy generation. To pass the test, the number of runs of length 3 (for samples of both 0 and 1) must be less than the programmed maximum value. The number of runs of length 3 must be greater than (maximum – range). If this test fails, the retry counter in SCMISC is decremented. If the retry count has not reached zero, a retry occurs. If the retry count has reached zero, an error is generated.

Note: This address (0x24) is only used as SCR3L when MCTL[PRGM] = 1. When MCTL[PRGM] = 0, this address is used as SCR3C readback (see [Section 45.4.17.2.18](#)) register.

Table 699. Statistical check run length 3 limit register - Offset: 2Ch

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved; always 0	-
30:16	RUN3_RNG	R/W	run length 3 range The number of runs of length 3 (for both 0 and 1) detected during entropy generation must be greater than RUN3_MAX – RUN3_RNG. If not, a retry or error occurs. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0
15	reserved	-	reserved; always 0	-
14:0	RUN3_MAX	R/W	run length 3 maximum limit Defines the maximum allowable runs of length 3 (for 0 and 1) detected during entropy generation. The number of runs of length 3 detected during entropy generation must be less than RUN3_MAX. If not, a retry or error occurs. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0

45.4.17.2.20 Statistical check run length 4 count register (SCR4C)

The statistical check run length 4 counters register is a read-only register used to read the final run length 4 counts after entropy generation. These counters start with the value in SCR4L[RUN4_MAX]. The R4_1_CT decrements each time a single one is sampled (preceded by a zero and followed by a zero). The R4_0_CT decrements each time a single zero is sampled (preceded by a one and followed by a one).

Note: This offset (0x24) is used as SCR4L when MCTL[PRGM] = 1 (see [Section 45.4.17.2.19](#)). When MCTL[PRGM] = 0, this offset is used as SCR4C readback register.

Table 700. Statistical check run length 4 count register - Offset: 30h

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved; always 0	-
30:16	R4_1_CT	R	runs of 1, length 4 count Reads the final runs of 1s, length 4 count after entropy generation. Requires MCTL[PRGM] = 0.	0
15	reserved	-	reserved; always 0	-
14:0	R4_0_CT	R	runs of 0, length 4 count Reads the final runs of 0s, length 4 count after entropy generation. Requires MCTL[PRGM] = 0.	0

45.4.17.2.21 Statistical check run length 4 limit register (SCR4L)

The statistical check run length 4 limit register defines the allowable maximum and minimum number of runs of length 4 detected during entropy generation. To pass the test, the number of runs of length 4 (for samples of both 0 and 1) must be less than the programmed maximum value. The number of runs of length 4 must be greater than (maximum – range). If this test fails, the retry counter in SCMISC is decremented. If the retry count has not reached zero, a retry occurs. If the retry count has reached zero, an error is generated.

Note: This address (0x24) is only used as SCR4L when MCTL[PRGM] = 1. When MCTL[PRGM] = 0, this address is used as SCR4C readback (see [Section 45.4.17.2.20](#)) register.

Table 701. Statistical check run length 4 limit register - Offset: 30h

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved; always 0	-
30:16	RUN4_RNG	R/W	run length 4 range The number of runs of length 4 (for both 0 and 1) detected during entropy generation must be greater than RUN4_MAX – RUN4_RNG. If not, a retry or error occurs. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0
15	reserved	-	reserved; always 0	-
14:0	RUN4_MAX	R/W	run length 4 maximum limit Defines the maximum allowable runs of length 4 (for 0 and 1) detected during entropy generation. The number of runs of length 4 detected during entropy generation must be less than RUN4_MAX. If not, a retry or error occurs. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0

45.4.17.2.22 Statistical check run length 5 count register (SCR5C)

The statistical check run length 5 counters register is a read-only register used to read the final run length 5 counts after entropy generation. These counters start with the value in SCR5L[RUN5_MAX]. The R5_1_CT decrements each time a single one is sampled (preceded by a zero and followed by a zero). The R5_0_CT decrements each time a single zero is sampled (preceded by a one and followed by a one).

Note: This offset (0x24) is used as SCR5L when MCTL[PRGM] = 1 (see [Section 45.4.17.2.23](#)). When MCTL[PRGM] = 0, this offset is used as SCR5C readback register.

Table 702. Statistical check run length 5 count register - Offset: 34h

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved; always 0	-
30:16	R5_1_CT	R	runs of 1, length 5 count Reads the final runs of 1s, length 5 count after entropy generation. Requires MCTL[PRGM] = 0.	0
15	reserved	-	reserved; always 0	-
14:0	R5_0_CT	R	runs of 0, length 5 count Reads the final runs of 0s, length 5 count after entropy generation. Requires MCTL[PRGM] = 0.	0

45.4.17.2.23 Statistical check run length 5 limit register (SCR5L)

The statistical check run length 5 limit register defines the allowable maximum and minimum number of runs of length 5 detected during entropy generation. To pass the test, the number of runs of length 5 (for samples of both 0 and 1) must be less than the programmed maximum value. The number of runs of length 5 must be greater than (maximum – range). If this test fails, the retry counter in SCMISC is decremented. If the retry count has not reached zero, a retry occurs. If the retry count has reached zero, an error is generated.

Note: This address (0x24) is only used as SCR5L when MCTL[PRGM] = 1. When MCTL[PRGM] = 0, this address is used as SCR5C readback (see [Section 45.4.17.2.22](#)) register.

Table 703. Statistical check run length 5 limit register - Offset: 34h

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved; always 0	-
30:16	RUN5_RNG	R/W	run length 5 range The number of runs of length 5 (for both 0 and 1) detected during entropy generation must be greater than RUN5_MAX – RUN5_RNG. If not, a retry or error occurs. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0
15	reserved	-	reserved; always 0	-
14:0	RUN5_MAX	R/W	run length 5 maximum limit Defines the maximum allowable runs of length 5 (for 0 and 1) detected during entropy generation. The number of runs of length 5 detected during entropy generation must be less than RUN5_MAX. If not, a retry or error occurs. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0

45.4.17.2.24 Statistical check run length 6+ count register (SCR6PC)

The statistical check run length 6+ counters register is a read-only register used to read the final run length 6+ counts after entropy generation. These counters start with the value in SCR6PL[RUN6P_MAX]. The R6P_1_CT decrements each time a single one is sampled (preceded by a zero and followed by a zero). The R6P_0_CT decrements each time a single zero is sampled (preceded by a one and followed by a one).

Note: This offset (0x24) is used as SCR6PL when MCTL[PRGM] = 1 (see [Section 45.4.17.2.25](#)). When MCTL[PRGM] = 0, this offset is used as SCR6PC readback register.

Table 704. Statistical check run length 6+ count register - Offset: 38h

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved; always 0	-
30:16	R6P_1_CT	R	runs of 1, length 6+ count Reads the final runs of 1s, length 6+ count after entropy generation. Requires MCTL[PRGM] = 0.	0
15	reserved	-	reserved; always 0	-
14:0	R6P_0_CT	R	runs of 0, length 6+ count Reads the final runs of 0s, length 6+ count after entropy generation. Requires MCTL[PRGM] = 0.	0

45.4.17.2.25 Statistical check run length 6+ limit register (SCR6PL)

The statistical check run length 6+ limit register defines the allowable maximum and minimum number of runs of length 6+ detected during entropy generation. To pass the test, the number of runs of length 6+ (for samples of both 0 and 1) must be less than the programmed maximum value. The number of runs of length 6+ must be greater than (maximum – range). If this test fails, the retry counter in SCMISC is decremented. If the retry count has not reached zero, a retry occurs. If the retry count has reached zero, an error is generated.

Note: This address (0x24) is only used as SCR6PL when MCTL[PRGM] = 1. When MCTL[PRGM] = 0, this address is used as SCR6PC readback (see [Section 45.4.17.2.24](#)) register.

Table 705. Statistical check run length 6+ limit register - Offset: 38h

Bit	Name	Access	Description and value	Reset value
31	reserved	-	reserved; always 0	-
30:16	RUN6P_RNG	R/W	run length 6+ range The number of runs of length 6+ (for both 0 and 1) detected during entropy generation must be greater than RUN6P_MAX – RUN6P_RNG. If not, a retry or error occurs. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0
15	reserved	-	reserved; always 0	-
14:0	RUN6P_MAX	R/W	run length 6+ maximum limit Defines the maximum allowable runs of length 6+ (for 0 and 1) detected during entropy generation. The number of runs of length 6+ detected during entropy generation must be less than RUN6P_MAX. If not, a retry or error occurs. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.	0

45.4.17.2.26 Status register (STATUS)

Various statistical tests are run as a normal part of the entropy generation process of the TRNG. The least-significant 16 bits of the STATUS register reflect the result of each of these tests. The status of these bits is valid when the TRNG has finished its entropy generation process. Software can determine when this occurs by polling the ENT_VAL bit in the miscellaneous control register.

Note: *There is a very small probability that a statistical test fails even though the TRNG operates properly. If this happens the TRNG automatically retries the entire entropy generation process, including running all the statistical tests. The value in RETRY_CT is decremented each time an entropy generation retry occurs. If a statistical check fails when the retry count is non-zero, a retry is initiated. However, if a statistical check fails when the retry count is zero, the RNG generates an error. By default RETRY_CT is initialized to 1. However, software can increase the retry count by writing to the RTY_CT field in the SCMISC register.*

If this register address is read while the RNG is in program mode (see PRGM field in MCTL register), all 0s are returned. If this register is read while the RNG is in run mode, the value returned is formatted as follows.

Table 706. Status register - Offset: 3Ch

Bit	Name	Access	Description and value	Reset value
31:20	reserved	-	reserved; always 0	-
19:16	RETRY_CT	R	retry count This represents the current number of entropy generation retries left before a statistical test failure causes the RNG to generate an error condition.	0
15	TFMB	R	test fail, monobit If TFMB = 1, the monobit test has failed.	0
14	TFP	R	test fail, poker If TFP = 1, the poker test has failed.	0
13	TFLR	R	test fail, long run If TFLR = 1, the long run test has failed.	0
12	TFSB	R	test fail, sparse bit If TFLR = 1, the sparse bit test has failed.	0
11	TF6PBR1	R	test fail, 6 plus bit run, sampling 1s If TF6PBR1 = 1, the 6 plus bit, sampling 1s test has failed.	0
10	TF6PBR0	R	test fail, 6 plus bit run, sampling 0s If TF6PBR0 = 1, the 6 plus bit, sampling 0s test has failed.	0
9	TF5BR1	R	test fail, 5-bit run, sampling 1s If TF5BR1 = 1, the 5-bit, sampling 1s test has failed.	0
8	TF5BR0	R	test fail, 5-bit run, sampling 0s If TF5BR0 = 1, the 5-bit, sampling 0s test has failed.	0
7	TF4BR1	R	test fail, 4-bit run, sampling 1s If TF4BR1 = 1, the 4-bit, sampling 1s test has failed.	0
6	TF4BR0	R	test fail, 4-bit run, sampling 0s If TF4BR0 = 1, the 4-bit, sampling 0s test has failed.	0
5	TF3BR1	R	test fail, 3-bit run, sampling 1s If TF3BR1 = 1, the 3-bit, sampling 1s test has failed.	0

Table 706. Status register - Offset: 3Ch...continued

Bit	Name	Access	Description and value	Reset value
4	TF3BR0	R	test fail, 3-bit run, sampling 0s If TF3BR0 = 1, the 3-bit, sampling 0s test has failed.	0
3	TF2BR1	R	test fail, 2-bit run, sampling 1s If TF2BR1 = 1, the 2-bit, sampling 1s test has failed.	0
2	TF2BR0	R	test fail, 2-bit run, sampling 0s If TF2BR0 = 1, the 2-bit, sampling 0s test has failed.	0
1	TF1BR1	R	test fail, 1-bit run, sampling 1s If TF1BR1 = 1, the 1-bit, sampling 1s test has failed.	0
0	TF1BR0	r	test fail, 1-bit run, sampling 0s If TF1BR0 = 1, the 0-bit, sampling 1s test has failed.	0

45.4.17.2.27 Entropy read register (ENTa)

The RNG TRNG can be programmed to generate an entropy value. To do this, set the MCTL[TRNG_ACC] bit to 1. Once the entropy value has been generated, the MCTL[ENT_VAL] bit will be set to 1. At this point, ENT0 through ENT15 may be read to retrieve the 512-bit entropy value. Note that once ENT15 is read, the entropy value will be cleared and a new value will begin generation, so it is important that ENT15 be read last. These registers are readable only when MCTL[PRGM] = 0 (Run Mode), MCTL[TRNG_ACC] = 1 (TRNG access mode) and MCTL[ENT_VAL] = 1. After at most one (1) bus clock cycle of reading a valid ENT15 register value, reading any ENT0 through ENT15 register would return zeros.

For a = 0 to 15

Table 707. Entropy read register - Offset: 40h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	ENT	R	entropy value Is only non-zero when MCTL[PRGM] = 0 (run mode) and MCTL[ENT_VAL] = 1 (entropy valid). The most significant bits of the entropy are read from the lowest offset. The least significant bits are read from the highest offset. Note: Reading the highest offset also clears the entire entropy value and starts a new entropy generation.	0

45.4.17.2.28 Statistical check poker count 1 and 0 register (PKRCNT10)

The statistical check poker count 1 and 0 register is a read-only register used to read the final poker test counts of 1h and 0h patterns. The poker 0h count increments each time a nibble of sample data is found to be 0h. The poker 1h count increments each time a nibble of sample data is found to be 1h.

Note: This register is only readable when $MCTL[PRGM] = 0$. Otherwise, zeroes are read.

Table 708. Statistical check poker count 1 and 0 register - Offset: 80h

Bit	Name	Access	Description and value	Reset value
31:16	PKR_1_CT	R	poker 1h count Total number of nibbles of sample data which were found to be 1h. Requires $MCTL[PRGM] = 0$.	0
15:0	PKR_0_CT	R	poker 0h count Total number of nibbles of sample data which were found to be 0h. Requires $MCTL[PRGM] = 0$.	0

45.4.17.2.29 Statistical check poker count 3 and 2 register (PKRCNT32)

The statistical check poker count 3 and 2 register is a read-only register used to read the final poker test counts of 3h and 2h patterns. The poker 2h count increments each time a nibble of sample data is found to be 2h. The poker 3h count increments each time a nibble of sample data is found to be 3h.

Note: This register is only readable when $MCTL[PRGM] = 0$. Otherwise, zeroes are read.

Table 709. Statistical check poker count 3 and 2 register - Offset: 84h

Bit	Name	Access	Description and value	Reset value
31:16	PKR_3_CT	R	poker 3h count Total number of nibbles of sample data which were found to be 3h. Requires $MCTL[PRGM] = 0$.	0
15:0	PKR_2_CT	R	poker 2h count Total number of nibbles of sample data which were found to be 2h. Requires $MCTL[PRGM] = 0$.	0

45.4.17.2.30 Statistical check poker count 5 and 4 register (PKRCNT54)

The statistical check poker count 5 and 4 register is a read-only register used to read the final poker test counts of 5h and 4h patterns. The poker 4h count increments each time a nibble of sample data is found to be 4h. The poker 5h count increments each time a nibble of sample data is found to be 5h.

Note: This register is only readable when $MCTL[PRGM] = 0$. Otherwise, zeroes are read.

Table 710. Statistical check poker count 5 and 4 register - Offset: 88h

Bit	Name	Access	Description and value	Reset value
31:16	PKR_5_CT	R	poker 5h count Total number of nibbles of sample data which were found to be 5h. Requires $MCTL[PRGM] = 0$.	0
15:0	PKR_4_CT	R	poker 4h count Total number of nibbles of sample data which were found to be 4h. Requires $MCTL[PRGM] = 0$.	0

45.4.17.2.31 Statistical check poker count 7 and 6 register (PKRCNT76)

The statistical check poker count 7 and 6 register is a read-only register used to read the final poker test counts of 7h and 6h patterns. The poker 6h count increments each time a nibble of sample data is found to be 6h. The poker 7h count increments each time a nibble of sample data is found to be 7h.

Note: This register is only readable when $MCTL[PRGM] = 0$. Otherwise, zeroes are read.

Table 711. Statistical check poker count 7 and 6 register - Offset: 8Ch

Bit	Name	Access	Description and value	Reset value
31:16	PKR_7_CT	R	poker 7h count Total number of nibbles of sample data which were found to be 7h. Requires $MCTL[PRGM] = 0$.	0
15:0	PKR_6_CT	R	poker 6h count Total number of nibbles of sample data which were found to be 6h. Requires $MCTL[PRGM] = 0$.	0

45.4.17.2.32 Statistical check poker count 9 and 8 register (PKRCNT98)

The statistical check poker count 9 and 8 register is a read-only register used to read the final poker test counts of 9h and 8h patterns. The poker 8h count increments each time a nibble of sample data is found to be 8h. The poker 9h count increments each time a nibble of sample data is found to be 9h.

Note: This register is only readable when $MCTL[PRGM] = 0$. Otherwise, zeroes are read.

Table 712. Statistical check poker count 9 and 8 register - Offset: 90h

Bit	Name	Access	Description and value	Reset value
31:16	PKR_9_CT	R	poker 9h count Total number of nibbles of sample data which were found to be 9h. Requires $MCTL[PRGM] = 0$.	0
15:0	PKR_8_CT	R	poker 8h count Total number of nibbles of sample data which were found to be 8h. Requires $MCTL[PRGM] = 0$.	0

45.4.17.2.33 Statistical check poker count B and A register (PKRCNTBA)

The statistical check poker count B and A register is a read-only register used to read the final poker test counts of Bh and Ah patterns. The poker Ah count increments each time a nibble of sample data is found to be Ah. The poker Bh count increments each time a nibble of sample data is found to be Bh.

Note: This register is only readable when $MCTL[PRGM] = 0$. Otherwise, zeroes are read.

Table 713. Statistical check poker count B and A register - Offset: 94h

Bit	Name	Access	Description and value	Reset value
31:16	PKR_B_CT	R	poker Bh count Total number of nibbles of sample data which were found to be Bh. Requires $MCTL[PRGM] = 0$.	0
15:0	PKR_A_CT	R	poker Ah count Total number of nibbles of sample data which were found to be Ah. Requires $MCTL[PRGM] = 0$.	0

45.4.17.2.34 Statistical check poker count D and C register (PKRCNTDC)

The statistical check poker count D and C register is a read-only register used to read the final poker test counts of Dh and Ch patterns. The poker Ch count increments each time a nibble of sample data is found to be Ch. The poker Dh count increments each time a nibble of sample data is found to be Dh.

Note: This register is only readable when $MCTL[PRGM] = 0$. Otherwise, zeroes are read.

Table 714. Statistical check poker count D and C register - Offset: 98h

Bit	Name	Access	Description and value	Reset value
31:16	PKR_D_CT	R	poker Dh count Total number of nibbles of sample data which were found to be Dh. Requires $MCTL[PRGM] = 0$.	0
15:0	PKR_C_CT	R	poker Ch count Total number of nibbles of sample data which were found to be Ch. Requires $MCTL[PRGM] = 0$.	0

45.4.17.2.35 Statistical check poker count F and E register (PKRCNTFE)

The statistical check poker count F and E register is a read-only register used to read the final poker test counts of Fh and Eh patterns. The poker Eh count increments each time a nibble of sample data is found to be Eh. The poker Fh count increments each time a nibble of sample data is found to be Fh.

Note: This register is only readable when $MCTL[PRGM] = 0$. Otherwise, zeroes are read.

Table 715. Statistical check poker count F and E register - Offset: 9Ch

Bit	Name	Access	Description and value	Reset value
31:16	PKR_F_CT	R	poker Fh count Total number of nibbles of sample data which were found to be Fh. Requires $MCTL[PRGM] = 0$.	0
15:0	PKR_E_CT	R	poker Eh count Total number of nibbles of sample data which were found to be Eh. Requires $MCTL[PRGM] = 0$.	0

45.4.17.2.36 Security configuration register (SEC_CFG)

The security configuration register is a read/write register used to control the test mode, programmability, and state modes of the TRNG. Many bits are placeholders for this version. More configurability will be added here. Clears on asynchronous system hard reset or POR.

Table 716. Security configuration register - Offset: A0h

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved	-
2	UNUSED2	R/W	This bit is unused; Ignore.	0

Table 716. Security configuration register - Offset: A0h...continued

Bit	Name	Access	Description and value	Reset value
1	NO_PRGM	R/W	If set, the TRNG registers cannot be programmed, regardless of the TRNG access mode in the TRNG miscellaneous control register.	0
			0b programability of registers only controlled by the access mode bit of the miscellaneous control register	
			1b overrides miscellaneous control register access mode and prevents TRNG register programming	
0	UNUSED0	R/W	This bit is unused; ignore.	0

45.4.17.2.37 Interrupt control register (INT_CTRL)

The interrupt control register is a read/write register used to control the status for the (currently) three important interrupts that are generated by the TRNG (see [Section 45.4.17.2.39](#)). Each interrupt can be cleared by deasserting the corresponding bit in the INT_CTRL register. Only a new interrupt reasserts the corresponding bit in the status register. Even if the interrupt is cleared or masked, interrupt status information can be read from the MCTL register.

Table 717. Interrupt control register - Offset: A4h

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved; always 0	-
2	FRQ_CT_FAIL	R/W	same behavior as bit 0 of this register	1
			0b same behavior as bit 0 of this register	
			1b same behavior as bit 0 of this register	
1	ENT_VAL	R/W	same behavior as bit 0 of this register	1
			0b same behavior as bit 0 of this register	
			1b same behavior as bit 0 of this register	
0	HW_ERR	R/W	bit position that can be cleared if corresponding bit of INT_STATUS register has been asserted.	1
			0b Corresponding bit of INT_STATUS register cleared	
			1b Corresponding bit of INT_STATUS register active	

45.4.17.2.38 Mask register (INT_MASK)

The interrupt mask register is a read/write register used to disable/mask the status reporting of the (currently) three important interrupts that are generated by the TRNG (see [Section 45.4.17.2.39](#)). Each interrupt can be masked/disabled by deasserting the corresponding bit in the INT_MASK register. Only setting this bit high reenables the interrupt in the status register. Even if the interrupt is cleared or masked, interrupt status information can be read from the MCTL register.

Table 718. Mask register - Offset: A8h

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved; always 0	-
2	FRQ_CT_FAIL	R/W	same behavior as bit 0 of this register	1
			0b same behavior as bit 0 of this register	
			1b same behavior as bit 0 of this register	
1	ENT_VAL	R/W	same behavior as bit 0 of this register	1
			0b same behavior as bit 0 of this register	
			1b same behavior as bit 0 of this register	
0	HW_ERR	R/W	bit position that can be cleared if corresponding bit of INT_STATUS register has been asserted.	1
			0b Corresponding bit of INT_STATUS register masked	
			1b Corresponding bit of INT_STATUS register active	

45.4.17.2.39 Interrupt status register (INT_STATUS)

The interrupt status register is a read register used to control and provide status for the (currently) three important interrupts that are generated by the TRNG. The TRNG interrupt signal alerts that TRNG has generated either a frequency count fail, an entropy valid, or an error interrupt. The cause of the interrupt can be decoded by checking the least significant bits of the INT_STATUS register. Each interrupt can be temporarily cleared by deasserting the corresponding bit in the INT_CTRL register. To mask the interrupts, clear the corresponding bits in the INT_MASK register (see [Section 45.4.17.2.38](#)). The description of each of the three interrupts is defined in the block guide under the MCTL register description. Even if the interrupt is cleared or masked, interrupt status information can be read from the MCTL register.

Table 719. Interrupt status register - Offset: ACh

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved; always 0	-
2	FRQ_CT_FAIL	R	frequency count fail The frequency counter has detected a failure. This may be due to improper programming of the FRQMAX and/or FRQMIN registers. Or a hardware failure in the ring oscillator.	0
			0b no hardware nor self test frequency errors	
			1b Frequency counter has detected a failure	
1	ENT_VAL	R	entropy valid Only asserts when TRNG ACC bit is set and then after an entropy value is generated. Is cleared when ENT15 is read. (ENT0 through ENT14 must be read before reading ENT15).	0
			0b busy generation entropy Any value is invalid.	
			1b TRNG can be stopped and entropy is valid if read	

Table 719. Interrupt status register - Offset: ACh...continued

Bit	Name	Access	Description and value	Reset value
0	HW_ERR	R/W	error status 1 = error detected. 0 = no error. Any HW error in the TRNG triggers this interrupt.	0
			0b no error	
			1b error detected	

45.4.17.2.40 Version ID register (MS) (VID1)

The version ID register (VID1) is an MSW read-only register used to identify the version of the TRNG in use. This register and the VID2 register must both be read to verify the expected version.

Table 720. Version ID register (MS) - Offset: F0h

Bit	Name	Access	Description and value	Reset value
31:16	IP_ID	R	shows the IP ID 0000_0000_0011_0000b - ID for TRNG	-
15:8	MAJ_REV	R	shows the major revision of the IP of the TRNG 0000_0001b - major revision number for TRNG	0
7:0	MIN_REV	R	show the minor revision of the IP of the TRNG 0000_0000b - Minor revision number for TRNG	0

45.4.17.2.41 Version ID register (LS) (VID2)

The version ID register (VID2) is an LSB read-only register used to identify the version of the TRNG in use. This register and the VID1 register must both be read to verify the expected version.

Table 721. Version ID register (LS) - Offset: F4h

Bit	Name	Access	Description and value	Reset value
31:24	ERA	R	shows the compile options for the TRNG 0000_0000b - COMPILE_OPT for TRNG	0
23:16	INTG_OPT	R	shows the integration options for the TRNG 0000_0000b - INTG_OPT for TRNG	0
15:8	ECO_REV	R	show the ECO revision of the IP of the TRNG 0000_0000b - TRNG_ECO_REV for TRNG	0
7:0	CONFIG_OPT	R	shows the configuration options of the IP of the TRNG 0000_0000b - TRNG_CONFIG_OPT for TRNG	

45.4.18 Physical unclonable function control (PUF)

45.4.18.1 PUF memory map

The PUF registers are located at base address: 4003_B000h.

Table 722. Physical unclonable function control registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CTRL	R/W	0h	PUF control register	32		Section 45.4.18.2.1
KEYINDEX	R/W	4h	PUF key index register	32		Section 45.4.18.2.2
KEYSIZE	R/W	8h	PUF key size register	32		Section 45.4.18.2.3
STAT	R	20h	PUF status register	32		Section 45.4.18.2.4
ALLOW	R/W	28h	PUF allow register	32		Section 45.4.18.2.5
KEYINPUT	W	40h	PUF key input register	32	0000_0000h	Section 45.4.18.2.6
CODEINPUT	W	44h	PUF code input register	32	0000_0000h	Section 45.4.18.2.7
CODEOUTPUT	R	48h	PUF code output register	32	0000_0000h	Section 45.4.18.2.8
KEYOUTINDEX	R	60h	PUF key output index register	32		Section 45.4.18.2.9
KEYOUTPUT	R	64h	PUF key output register	32	0000_0000h	Section 45.4.18.2.10
IFSTAT	R/W	DCh	PUF interface status and clear register	32		Section 45.4.18.2.11
INTEN	R/W	100h	PUF interrupt enable	32		Section 45.4.18.2.12
INTSTAT	R/W	104h	PUF interrupt status	32		Section 45.4.18.2.13
CFG	R/W	10Ch	PUF config register for block bits	32		Section 45.4.18.2.14
KEYLOCK	R/W	200h	key lock	32		Section 45.4.18.2.15
KEYENABLE	R/W	204h	key enable	32		Section 45.4.18.2.16
KEYRESET	W	208h	reinitialize keys shift registers counters	32	0000_0000h	Section 45.4.18.2.17
ISXBLK	W	20Ch	block index	32	AAAA_AAAAh	Section 45.4.18.2.18
IDXBLK_DP	W	210h	block index duplicate	32	AAAA_AAAAh	Section 45.4.18.2.19
KEYMASK0 - KEYMASK4	W	214h - 224h	key mask	32	0000_0000h	Section 45.4.18.2.20
IDXBLK_STATUS	R	254h	block index status	32	0000_0000h	Section 45.4.18.2.21
SHIFT_STATUS	R	258h	key shift status	32		Section 45.4.18.2.22

45.4.18.2 Register descriptions

45.4.18.2.1 PUF control register (CTRL)

Table 723. PUF control register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:7	reserved	-	reserved	-
6	GETKEY	R/W	begin get key operation	0
5	reserved	-	reserved	-
4	SETKEY	R/W	begin set user key operation	0
3	GENERATEKEY	R/W	begin set intrinsic key operation	0
2	START	R/W	begin start operation	0
1	ENROLL	R/W	begin enroll operation	0
0	ZEROIZE	R/W	begin zeroize operation for PUF and enter the error state	0

45.4.18.2.2 PUF key index register (KEYINDEX)

Table 724. PUF key index register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved	-
3:0	KEYIDX	R/W	key index for set key operations	0

45.4.18.2.3 PUF key size register (KEYSIZE)

Table 725. PUF key size register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved	-
5:0	KEYSIZE	R/W	key size for set key operations	0

45.4.18.2.4 PUF status register (STAT)

Table 726. PUF status register - Offset: 20h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved	-
7	CODEOUTAVAIL	R	next part of AC/KC is available	0
6	CODEINREQ	R	request for next part of AC/KC	0
5	KEYOUTAVAIL	R	next part of key is available	0
4	KEYINREQ	R	request for next part of key	0
3	reserved	-	reserved	-
2	ERROR	R	PUF is in the Error state; no operations can be performed	0

Table 726. PUF status register - Offset: 20h...continued

Bit	Name	Access	Description and value	Reset value
1	SUCCESS	R	last operation was successful	0
0	BUSY	R	indicates that operation is in progress	1

45.4.18.2.5 PUF allow register (ALLOW)

Table 727. PUF allow register - Offset: 28h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved	-
7	allow_bist	R/W	BIST is allowed to be started	0
6:4	reserved	-	reserved	-
3	ALLOWGETKEY	R	get key operation is allowed	0
2	ALLOWSETKEY	R	set key operations are allowed	0
1	ALLOWSTART	R	start operation is allowed	0
0	ALLOWENROLL	R	enroll operation is allowed	0

45.4.18.2.6 PUF key input register (KEYINPUT)

Table 728. PUF key input register - Offset: 40h

Bit	Name	Access	Description and value	Reset value
31:0	KEYIN	W	key input data	0

45.4.18.2.7 PUF code input register (CODEINPUT)

Table 729. PUF code input register - Offset: 44h

Bit	Name	Access	Description and value	Reset value
31:0	CODEIN	W	AC/KC input data	0

45.4.18.2.8 PUF code output register (CODEOUTPUT)

Table 730. PUF code output register - Offset: 48h

Bit	Name	Access	Description and value	Reset value
31:0	CODEOUT	R	AC/KC output data	0

45.4.18.2.9 PUF key output index register (KEYOUTINDEX)

Table 731. PUF key output index register - Offset: 60h

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved	-

Table 731. PUF key output index register - Offset: 60h...continued

Bit	Name	Access	Description and value	Reset value
3:0	KEYOUTIDX	R	key index for the key that is currently output via the key output register	0

45.4.18.2.10 PUF key output register (KEYOUTPUT)

Table 732. PUF key output register - Offset: 64h

Bit	Name	Access	Description and value	Reset value
31:0	KEYOUT	R	key output data	0

45.4.18.2.11 PUF interface status and clear register (IFSTAT)

Table 733. PUF interface status and clear register - Offset: DCCh

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved	-
0	ERROR	R/W	Indicates that an APB error has occurred Writing logic1 clears the IF_ERROR bit.	0

45.4.18.2.12 PUF interrupt enable (INTEN)

Table 734. PUF interrupt enable - Offset: 100h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved	-
7	CODEOUTAVAILEN	R/W	enable corresponding interrupt. Note that bit numbers match those assigned in QK_SR (Quiddikey status register) Note: Bit numbers match the numbers assigned in QK_SR (quiddikey status register)	0
6	CODEINREQEN	R/W	enable corresponding interrupt Note: Bit numbers match the numbers assigned in QK_SR (quiddikey status register)	0
5	KEYOUTAVAILEN	R/W	enable corresponding interrupt Note: Bit numbers match the numbers assigned in QK_SR (quiddikey status register)	0
4	KEYINREQEN	R/W	enable corresponding interrupt Note: Bit numbers match the numbers assigned in QK_SR (quiddikey status register)	0
3	reserved	-	reserved	-
2	ERROREN	R/W	enable corresponding interrupt Note: Bit numbers match the numbers assigned in QK_SR (quiddikey status register)	0
1	SUCCESSEN	R/W	enable corresponding interrupt Note: Bit numbers match the numbers assigned in QK_SR (quiddikey status register)	0

Table 734. PUF interrupt enable - Offset: 100h...continued

Bit	Name	Access	Description and value	Reset value
0	READYEN	R/W	enable corresponding interrupt Note: Bit numbers match the numbers assigned in QK_SR (quiddikey status register)	0

45.4.18.2.13 PUF interrupt status (INTSTAT)

Table 735. PUF interrupt status - Offset: 104h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved	-
7	CODEOUTAVAIL	R/W	level sensitive interrupt Cleared when interrupt source clears.	0
6	CODEINREQ	R/W	level sensitive interrupt Cleared when interrupt source clears.	0
5	KEYOUTAVAIL	R/W	level sensitive interrupt Cleared when interrupt source clears.	0
4	KEYINREQ	R/W	level sensitive interrupt Cleared when interrupt source clears.	0
3	reserved	-	reserved	-
2	ERROR	R/W	level sensitive interrupt Cleared when interrupt source clears.	0
1	SUCCESS	R/W	level sensitive interrupt Cleared when interrupt source clears.	0
0	READY	R/W	Triggers on falling edge of busy Write 1 to clear.	0

45.4.18.2.14 PUF config register for block bits (CFG)

Table 736. PUF config register for block bits - Offset: 10Ch

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1	BLOCKKEYOUTPUT	R/W	block set key operation Write 1 to set; cleared on reset.	0
0	BLOCKENROLL_SETKEY	R/W	block enroll operation Write 1 to set; cleared on reset.	0

45.4.18.2.15 Key lock (KEYLOCK)

Table 737. Key lock - Offset: 200h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved	-
9:8	KEY4	R/W	key 4 Note: When this field is written with a value different from '10', its value can no longer be modified until a power-on reset occurs.	1
			10 Write access to KEY4MASK, KEYENABLE.KEY4, and KEYRESET.KEY4 is allowed.	
			00; 01; 11 Write access to KEY4MASK, KEYENABLE.KEY4, and KEYRESET.KEY4 is NOT allowed.	
7:6	KEY3	R/W	key 3 Note: When this field is written with a value different from '10', its value can no longer be modified until a power-on reset occurs..	1
			10 Write access to KEY3MASK, KEYENABLE.KEY3, and KEYRESET.KEY3 is allowed.	
			00; 01; 11 Write access to KEY3MASK, KEYENABLE.KEY3, and KEYRESET.KEY3 is NOT allowed.	
5:4	KEY2	R/W	key 2 Note: When this field is written with a value different from '10', its value can no longer be modified until a power-on reset occurs..	1
			10 Write access to KEY2MASK, KEYENABLE.KEY2, and KEYRESET.KEY2 is allowed.	
			00; 01; 11 Write access to KEY2MASK, KEYENABLE.KEY2, and KEYRESET.KEY2 is NOT allowed.	
3:2	KEY1	R/W	key 1 Note: When this field is written with a value different from '10', its value can no longer be modified until a power-on reset occurs..	1
			10 Write access to KEY1MASK, KEYENABLE.KEY1, and KEYRESET.KEY1 is allowed.	
			00; 01; 11 Write access to KEY1MASK, KEYENABLE.KEY1, and KEYRESET.KEY1 is NOT allowed.	
1:0	KEY0	R/W	key 0 Note: When this field is written with a value different from '10', its value can no longer be modified until a power-on reset occurs..	1
			10 Write access to KEY0MASK, KEYENABLE.KEY0, and KEYRESET.KEY0 is allowed.	
			00; 01; 11 Write access to KEY0MASK, KEYENABLE.KEY0, and KEYRESET.KEY0 is NOT allowed.	

45.4.18.2.16 Key enable (KEYENABLE)

Table 738. Key enable - Offset: 204h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved	-
9:8	KEY4	R/W	key 4	0
			10 Data coming out from PUF index 0 interface are shifted in KEY4 register.	
			00; 01; 11 Data coming out from PUF Index 0 interface are NOT shifted in KEY4 register.	
7:6	KEY3	R/W	key 3	0
			10 Data coming out from PUF index 0 interface are shifted in KEY3 register.	
			00; 01; 11 Data coming out from PUF Index 0 interface are NOT shifted in KEY3 register.	
5:4	KEY2	R/W	key 2	0
			10 Data coming out from PUF index 0 interface are shifted in KEY2 register.	
			00; 01; 11 Data coming out from PUF Index 0 interface are NOT shifted in KEY2 register.	
3:2	KEY1	R/W	key 1	0
			10 Data coming out from PUF index 0 interface are shifted in KEY1 register.	
			00; 01; 11 Data coming out from PUF Index 0 interface are NOT shifted in KEY1 register.	
1:0	KEY0	R/W	key 0	0
			10 Data coming out from PUF index 0 interface are shifted in KEY0 register.	
			00; 01; 11 Data coming out from PUF Index 0 interface are NOT shifted in KEY0 register.	

45.4.18.2.17 Reinitialize keys shift registers counters (KEYRESET)

Table 739. Reinitialize keys shift registers counters - Offset: 208h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved	-
9:8	KEY4	R/W	key 4	0
			10 reset KEY4 shift register Self clearing; must be done before loading any new key.	
7:6	KEY3	R/W	key 3	0
			10 reset KEY3 shift register Self clearing; must be done before loading any new key.	

Table 739. Reinitialize keys shift registers counters - Offset: 208h...continued

Bit	Name	Access	Description and value	Reset value
5:4	KEY2	R/W	key 2	0
			10 reset KEY2 shift register Self clearing; must be done before loading any new key.	
3:2	KEY1	R/W	key 1	0
			10 reset KEY1 shift register Self clearing; must be done before loading any new key.	
1:0	KEY0	R/W	key 0	0
			10 reset KEY0 shift register Self clearing; must be done before loading any new key.	

45.4.18.2.18 Block index (IDXBLK)

Table 740. Block index - Offset: 20Ch

Bit	Name	Access	Description and value	Reset value
31:30	IDX15	W	use to block PUF index 15	1
29:28	IDX14	W	use to block PUF index 14	1
27:26	IDX13	W	use to block PUF index 13	1
25:24	IDX12	W	use to block PUF index 12	1
23:22	IDX11	W	use to block PUF index 11	1
21:20	IDX10	W	use to block PUF index 10	1
19:18	IDX9	W	use to block PUF index 9	1
17:16	IDX8	W	use to block PUF index 8	1
15:14	IDX7	W	use to block PUF index 7	1
13:12	IDX6	W	use to block PUF index 6	1
11:10	IDX5	W	use to block PUF index 5	1
9:8	IDX4	W	use to block PUF index 4	1
7:6	IDX3	W	use to block PUF index 3	1
5:4	IDX2	W	use to block PUF index 2	1
3:2	IDX1	W	use to block PUF index 1	1
1:0	IDX0	W	use to block PUF index 0	1

45.4.18.2.19 Block index duplicate (IDXBLK_DP)

Table 741. Block index duplicate - Offset: 210h

Bit	Name	Access	Description and value	Reset value
31:30	IDX15	W	use to block PUF index 15	1
29:28	IDX14	W	use to block PUF index 14	1

Table 741. Block index duplicate - Offset: 210h...continued

Bit	Name	Access	Description and value	Reset value
27:26	IDX13	W	use to block PUF index 13	1
25:24	IDX12	W	use to block PUF index 12	1
23:22	IDX11	W	use to block PUF index 11	1
21:20	IDX10	W	use to block PUF index 10	1
19:18	IDX9	W	use to block PUF index 9	1
17:16	IDX8	W	use to block PUF index 8	1
15:14	IDX7	W	use to block PUF index 7	1
13:12	IDX6	W	use to block PUF index 6	1
11:10	IDX5	W	use to block PUF index 5	1
9:8	IDX4	W	use to block PUF index 4	1
7:6	IDX3	W	use to block PUF index 3	1
5:4	IDX2	W	use to block PUF index 2	1
3:2	IDX1	W	use to block PUF index 1	1
1:0	IDX0	W	use to block PUF index 0	1

45.4.18.2.20 Key mask (KEYMASK0 - KEYMASK4)

Table 742. Offset

Register	Offset
KEYMASK0	214h
KEYMASK1	218h
KEYMASK2	21Ch
KEYMASK3	220h
KEYMASK4	224h

Table 743. Key mask

Bit	Name	Access	Description and value	Reset value
31:0	KEYMASK	W	This register 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 if there is a full IC reset.	0

45.4.18.2.21 Block index status (IDXBLK_STATUS)

Table 744. Block index status - Offset: 254h

Bit	Name	Access	Description and value		Reset value
31:30	IDX15	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	
29:28	IDX14	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	
27:26	IDX13	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	
25:24	IDX12	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	
23:22	IDX11	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	
21:20	IDX10	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	
19:18	IDX9	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	
17:16	IDX8	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	

Table 744. Block index status - Offset: 254h...continued

Bit	Name	Access	Description and value		Reset value
15:14	IDX7	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	
13:12	IDX6	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	
11:10	IDX5	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	
9:8	IDX4	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	
7:6	IDX3	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	
5:4	IDX2	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	
3:2	IDX1	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	
1:0	IDX0	R	status block index		0x2
			0x1	PUF index is blocked	
			0x2	PUF index is accessible	
			others	reserved	

45.4.18.2.22 Key shift status (SHIFT_STATUS)

This register describes number of words loaded into the key hold register. The user can rely on this register to ensure that, before enabling encryption/decryption, the correct number of words is loaded for a given crypto

engine. PRINCE requires a 128-bit secret key. So, before starting PRINCE operation, four 32-bit words must be loaded. For AES128 four words must be loaded as well. For AES192, six words must be loaded. For AES256, before starting AES operation, 8 words must be loaded.

Table 745. Key shift status - Offset: 258h

Bit	Name	Access	Description and value	Reset value
31:20	reserved	-	reserved	-
19:16	KEY4	R	index counter from key 4 shift register	0
15:12	KEY3	R	index counter from key 3 shift register	0
11:8	KEY2	R	index counter from key 2 shift register	0
7:4	KEY1	R	index counter from key 1 shift register	0
3:0	KEY0	R	index counter from key 0 shift register	0

45.4.19 PUF SRAM control

45.4.19.1 PUF SRAM control memory map

The PUF_SRAM_CTRL registers are located at base address: 4003_B000h.

Table 746. PUF SRAM control registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CFG	R/W	300h	configuration register	32		Section 45.4.19.2.1
STATUS	W	304h	status register	32		Section 45.4.19.2.2
SRAM_CFG	R/W	308h	SRAM configuration register	32		Section 45.4.19.2.3
INT_CLR_ENABLE	W	3D8h	interrupt enable clear register	32		Section 45.4.19.2.4
INT_SET_ENABLE	W	3DCh	interrupt enable set register	32		Section 45.4.19.2.5
INT_STATUS	W	3E0h	interrupt status register	32		Section 45.4.19.2.6
INT_ENABLE	W	3E4h	interrupt enable register	32		Section 45.4.19.2.7
INT_CLR_STATUS	W	3E8h	interrupt status clear register	32		Section 45.4.19.2.8
INT_SET_STATUS	W	3ECh	interrupt status set register	32		Section 45.4.19.2.9
MODULE_ID	R	3FCh	module identification register	32		Section 45.4.19.2.10

45.4.19.2 Register descriptions

45.4.19.2.1 Configuration register (CFG)

Table 747. Configuration register - Offset: 300h

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved	-
2	CKGATING	R/W	PUF SRAM clock gating control	0
			0 disabled	
			1 enabled	
1	MODE	R/W	PUF SRAM controller operating mode	0
			0 normal mode SRAM discharge managed by hardware.	
			1 backup mode SRAM discharge time-managed by software.	
0	ENABLE	R/W	PUF SRAM controller activation	0
			0 disabled	
			1 enabled	

45.4.19.2.2 Status register (STATUS)

Table 748. Status register - Offset: 304h

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved	-
0	READY	R	PUF SRAM controller state	0
			0 not ready PUF cannot access PUF SRAM.	
			1 ready PUF can access PUF SRAM.	

45.4.19.2.3 SRAM configuration register (SRAM_CFG)

Table 749. SRAM configuration register - Offset: 308h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved	-
15:13	WAM	R/W	SRAM write-assist settings	1
12:11	RAM	R/W	SRAM read-assist settings	0
10:8	WM	R/W	SRAM write-margin control settings	1
7:3	RM	R/W	SRAM read-margin control settings	0
2:0	SMB	R/W	SRAM source-biasing voltage control	0

45.4.19.2.4 Interrupt enable clear register (INT_CLR_ENABLE)

Table 750. Interrupt enable clear register - Offset: 3D8h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1	APB_ERR	W	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.	
0	READY	W	READY interrupt enable clear	0
			0 no effect	
			1 clears the READY bit field in register INT_ENABLE Automatically reset by the Hardware.	

45.4.19.2.5 Interrupt enable set register (INT_SET_ENABLE)

Table 751. Interrupt enable set register - Offset: 3DCh

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1	APB_ERR	W	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.	
0	READY	W	READY interrupt enable set	0
			0 no effect	
			1 sets the READY bit field in register INT_ENABLE Automatically reset by the Hardware.	

45.4.19.2.6 Interrupt status register (INT_STATUS)

Table 752. Interrupt status register - Offset: 3E0h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1	APB_ERR	W	APB_ERR interrupt status Set when an access to registers has been denied in application of the access security rules determined by pprot,pprot_mask, and pprot_match.	0
			0 not pending	
			1 pending	
0	READY	W	READY interrupt status Set at the end of the SRAM initialization phase (whatever the value of the READY bit in INT_ENABLE register), indicating that PUF (Quiddykey) can use the SRAM. An interrupt is only generated when this bit and the READY bit in INT_ENABLE register are set.	0
			0 no pending	
			1 pending	

45.4.19.2.7 Interrupt enable register (INT_ENABLE)

Table 753. Interrupt enable register - Offset: 3E4h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1	APB_ERR	W	APB_ERR interrupt enable	0
			0 disabled	
			1 enabled	
0	READY	W	READY interrupt enable	0
			0 disabled	
			1 enabled	

45.4.19.2.8 Interrupt status clear register (INT_CLR_STATUS)

Table 754. Interrupt status clear register - Offset: 3E8h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1	APB_ERR	W	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.	
0	READY	W	READY interrupt status clear	0
			0 no effect	
			1 clears the READY bit field in register INT_STATUS Automatically reset by the Hardware.	

45.4.19.2.9 Interrupt status set (INT_SET_STATUS)

Table 755. Interrupt status set - Offset: 3ECh

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1	APB_ERR	W	APB_ERR interrupt status set	0
			0 no effect	
			1 sets the APB_ERR bit field in register INT_STATUS Automatically reset by the Hardware.	
0	READY	W	READY interrupt status set	0
			0 no effect	
			1 sets the READY bit field in register INT_STATUS Automatically reset by the Hardware.	

45.4.19.2.10 Module identification register (MODULE_ID)

Table 756. Module identification register - Offset: 3FCh

Bit	Name	Access	Description and value	Reset value
31:16	ID	R	identification number Unique module identifier indicating the module.	1
15:12	MAJOR_REV	R	major revision of module implementation Starting at 0. There is no guaranteed software compatibility between major revisions.	0
11:8	MINOR_REV	R	minor revision of module implementation Starting at 0. There is always software compatibility between minor revisions.	0
7:0	APERTURE	R	aperture size of the register interface It is encoded as (size / 256byte) – 1. So, 0x00 means 256 bytes	0

45.4.20 Programmable logic unit (PLU)

45.4.20.1 PLU memory map

The PLU registers are located at base address: 4003_D000h.

Table 757. Programmable logic unit control registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
LUT0_INP0_MUX - LUT25_INP4_MUX	R/W	0h - 330h	input select register for LUTn (0 to 25), Inputx (0 to 4)	32		Section 45.4.20.2.1
LUT0_TRUTH - LUT25_TRUTH	R/W	800h - 864h	truth table contents for LUTn	32	0000_0000h	Section 45.4.20.2.2
OUTPUTS	R/W	900h	provides the current state of the 8 designated PLU outputs	32	0000_0000h	Section 45.4.20.2.3
WAKEINT_CTRL	R/W	904h	wake-up interrupt control for PLU	32	0000_0000h	Section 45.4.20.2.4
OUTPUT0_MUX - OUTPUT7_MUX	R/W	C00h - C1Ch	selects the source to be connected to PLU output n	32	0000_0000h	Section 45.4.20.2.5

45.4.20.2 Register descriptions

45.4.20.2.1 LUTn input y MUX (LUTn_INPy_MUX)

- For n = 0 to 25
- For y = 0 to 4

Table 758. LUTn input y MUX - Offset: 0h + (n × 20h) + (y × 4h)

Bit	Name	Access	Description and value	Reset value																														
31:6	reserved	-	reserved Software must not write ones to reserved bits.	-																														
5:0	LUTn_INPy	R/W	selects the input source to be connected to LUTn input y For each LUT, the slot associated with the output from LUTn itself is tied low. <table><tr><td>00_0000b</td><td>PLU primary inputs 0</td></tr><tr><td>00_0001b</td><td>PLU primary inputs 1</td></tr><tr><td>00_0010b</td><td>PLU primary inputs 2</td></tr><tr><td>00_0011b</td><td>PLU primary inputs 3</td></tr><tr><td>00_0100b</td><td>PLU primary inputs 4</td></tr><tr><td>00_0101b</td><td>PLU primary inputs 5</td></tr><tr><td>00_0110b</td><td>output of LUT0</td></tr><tr><td>00_0111b</td><td>output of LUT1</td></tr><tr><td>00_1000b</td><td>output of LUT2</td></tr><tr><td>00_1001b</td><td>output of LUT3</td></tr><tr><td>00_1010b</td><td>output of LUT4</td></tr><tr><td>00_1011b</td><td>output of LUT5</td></tr><tr><td>00_1100b</td><td>output of LUT6</td></tr><tr><td>00_1101b</td><td>output of LUT7</td></tr><tr><td>00_1110b</td><td>output of LUT8</td></tr></table>	00_0000b	PLU primary inputs 0	00_0001b	PLU primary inputs 1	00_0010b	PLU primary inputs 2	00_0011b	PLU primary inputs 3	00_0100b	PLU primary inputs 4	00_0101b	PLU primary inputs 5	00_0110b	output of LUT0	00_0111b	output of LUT1	00_1000b	output of LUT2	00_1001b	output of LUT3	00_1010b	output of LUT4	00_1011b	output of LUT5	00_1100b	output of LUT6	00_1101b	output of LUT7	00_1110b	output of LUT8	0
00_0000b	PLU primary inputs 0																																	
00_0001b	PLU primary inputs 1																																	
00_0010b	PLU primary inputs 2																																	
00_0011b	PLU primary inputs 3																																	
00_0100b	PLU primary inputs 4																																	
00_0101b	PLU primary inputs 5																																	
00_0110b	output of LUT0																																	
00_0111b	output of LUT1																																	
00_1000b	output of LUT2																																	
00_1001b	output of LUT3																																	
00_1010b	output of LUT4																																	
00_1011b	output of LUT5																																	
00_1100b	output of LUT6																																	
00_1101b	output of LUT7																																	
00_1110b	output of LUT8																																	

Table 758. LUTn input y MUX - Offset: $0h + (n \times 20h) + (y \times 4h)$...continued

Bit	Name	Access	Description and value		Reset value
5:0 (cont'd)	LUTn_INPy	R/W	00_1111b	output of LUT9	0
			01_0000b	output of LUT10	
			01_0001b	output of LUT11	
			01_0010b	output of LUT12	
			01_0011b	output of LUT13	
			01_0100b	output of LUT14	
			01_0101b	output of LUT15	
			01_0110b	output of LUT16	
			01_0111b	output of LUT17	
			01_1000b	output of LUT18	
			01_1001b	output of LUT19	
			01_1010b	output of LUT20	
			01_1011b	output of LUT21	
			01_1100b	output of LUT22	
			01_1101b	output of LUT23	
			01_1110b	output of LUT24	
			01_1111b	output of LUT25	
			10_0000b	state(0)	
			10_0001b	state(1)	
			10_0010b	state(2)	
			10_0011b	state(3)	

45.4.20.2.2 Truth table contents for LUTn (LUTn_TRUTH)

For $n = 0$ to 25

Table 759. Truth table contents for LUTn - Offset: $800h + (n \times 20h)$

Bit	Name	Access	Description and value	Reset value
31:6	LUTn_TRUTH	R/W	Specifies the truth table contents for LUTn	0

45.4.20.2.3 Current state of the 8 designated PLU outputs (OUTPUTS)

Table 760. Current state of the 8 designated PLU outputs - Offset: 900h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Software must not write ones to reserved bits.	-
7:0	OUTPUT_STATE	R	provides the current state of the 8 designated PLU outputs	0

45.4.20.2.4 Wake-up interrupt control for PLU (WAKEINT_CTRL)

Table 761. Wake-up interrupt control for PLU - Offset: 904h

Bit	Name	Access	Description and value	Reset value								
31:14	reserved	-	reserved Software must not write ones to reserved bits.	-								
13	INTR_CLEAR	R/W	write to clear wakeint_latched When using the registered wake-up/interrupt option (LATCH_ENA = '1'), writing a '1' to this bit clears the wake-up/interrupt request flag. Writing a '0' to this bit has no effect. Before this write-to-clear takes effect, a delay of up to 1.5 plu_clkin clock cycles occurs. At that point, this bit is also cleared. To determine when the wake-up/interrupt request has been removed, software can poll this bit. 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								
12	LATCH_ENABLE	R/W	latch the interrupt, then can be cleared with next bit INTR_CLEAR Latch the interrupt and then it can be cleared with next bit INTR_CLEAR. Setting this bit specifies the 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 the use of the glitch filter. If this bit is set, the FILTER_MODE field must be set to "00" (bypass mode). If this bit is set, the wake-up/interrupt request is set on the rising-edge of plu_clkin when the raw wake-up/interrupt signal is high. The request must be cleared using software.	0								
11:10	FILTER_CLKSEL	R/W	HCLK is divided by 2**filter_clkssel Note: These clocking options are chip-dependent. <table><tr><td>00b</td><td>selects the 1 MHz low-power oscillator as the filter clock</td></tr><tr><td>01b</td><td>selects the 12 Mhz FRO as the filter clock</td></tr><tr><td>10b</td><td>if provided, selects a third filter clock source</td></tr><tr><td>11b</td><td>reserved</td></tr></table>	00b	selects the 1 MHz low-power oscillator as the filter clock	01b	selects the 12 Mhz FRO as the filter clock	10b	if provided, selects a third filter clock source	11b	reserved	0
00b	selects the 1 MHz low-power oscillator as the filter clock											
01b	selects the 12 Mhz FRO as the filter clock											
10b	if provided, selects a third filter clock source											
11b	reserved											

Table 761. Wake-up interrupt control for PLU - Offset: 904h...continued

Bit	Name	Access	Description and value	Reset value
9:8	FILTER_MODE	R/W	control input of the PLU; add filtering for glitch	0
			00b bypass mode; filtering is disabled The raw wake-up/interrupt will be passed-through.	
			01b filter 1 clock period Any pulse duration shorter than one cycle of the designated filter clock is filtered out. Pulse widths up to two cycles long may be filtered.	
			10b filter 2 clock period Any pulse duration shorter than two cycles of the designated filter clock is filtered out. Pulse widths up to three cycles long may be filtered.	
			11b filter 3 clock period Any pulse duration shorter than three cycles of the designated filter clock is filtered out. Pulse widths up to four cycles long may be filtered.	
7:0	MASK	R/W	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 the wake-up/interrupt generation. To generate the wake-up/interrupt request, all enabled PLU outputs are OR-ed. A '0' in any bit of this register blocks the corresponding PLU output from causing a wake-up/interrupt.	0

45.4.20.2.5 Selection of source to be connected to PLU output n (OUTPUTn)

For n = 0 to 7.

Table 762. Selection of source to be connected to PLU output n - Offset: 0h + (n × 4h)

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved Software must not write ones to reserved bits.	-
4:0	OUTPUTn	R/W	selects the source to be connected to PLU output n	0
			0_0000b PLU output 0	
			0_0001b PLU output 1	
			0_0010b PLU output 2	
			0_0011b PLU output 3	
			0_0100b PLU output 4	
			0_0101b PLU output 5	
			0_0110b PLU output 6	
			0_0111b PLU output 7	
			0_1000b PLU output 8	
			0_1001b PLU output 9	
			0_1010b PLU output 10	

Table 762. Selection of source to be connected to PLU output n - Offset: 0h + (n × 4h)...continued

Bit	Name	Access	Description and value		Reset value
4:0 (cont'd)	OUTPUTn	R/W	0_1011b	PLU output 11	0
			0_1100b	PLU output 12	
			0_1101b	PLU output 13	
			0_1110b	PLU output 14	
			0_1111b	PLU output 15	
			1_0000b	PLU output 16	
			1_0001b	PLU output 17	
			1_0010b	PLU output 18	
			1_0011b	PLU output 19	
			1_0100b	PLU output 20	
			1_0101b	PLU output 21	
			1_0110b	PLU output 22	
			1_0111b	PLU output 23	
			1_1000b	PLU output 24	
			1_1001b	PLU output 25	
			1_1010b	state(0)	
			1_1011b	state(1)	
			1_1100b	state(2)	
			1_1101b	state(3)	

45.4.21 ROM control and patch (ROMCP)

45.4.21.1 ROMCP memory map

The ROMCP registers are located at base address: 4003_E000h.

Table 763. ROM control and patch control registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
ROMPATCH7D - ROMPATCH0D	R/W	D4h - F0h	ROMCP data registers	32	0000_0000h	Section 45.4.21.2.1
ROMPATCHCNTL	R/W	F4h	ROMCP control register	32	0840_0000h	Section 45.4.21.2.2
ROMPATCHENH	R/W	F8hh	ROMCP enable register high	32	0000_0000h	Section 45.4.21.2.3
ROMPATCHENL	R/W	FC	ROMCP enable register low	32	0000_0000h	Section 45.4.21.2.4
ROMPATCH0A - ROMPATCH15A	R/W	100h - 13Ch	ROMCP address registers	32	0000_0000h	Section 45.4.21.2.5
ROMPATCHSR	W1C	208h	ROMCP status register	32	0000_0000h	Section 45.4.21.2.6

45.4.21.2 Register descriptions

45.4.21.2.1 ROMCP data registers (ROMPATCHaD)

The ROMCP data registers (ROMCP_ROMPATCH7D through ROMCP_ROMPATCH0D) store the data to use for the 8 1-word data fix events. Each register is associated with an address comparator (7 through 0). When a data fixing event occurs, the value in the data register corresponding to the comparator that has the address match is put on the romcp_hrdata[31:0] bus until romcp_hready is the ROM controller asserts to terminate the access. A MUX external to the ROMCP selects this data over that of romcp_hrdata[31:0] in returning read data to the Arm core. The selection is done with the control bus rompatch_romcp_hrdata_ovr[1:0] with both bits asserted by the ROMCP.

If more than one address comparator matches, the highest-numbered one takes precedence. The value in corresponding data register is used for the patching event.

For a = 0 to 7.

Table 764. ROMCP data registers - Offset: F0h + (n × -4h)

Bit	Name	Access	Description and value	Reset value
31:0	DATAx	R/W	data fix registers; stores the data used for 1-word data fix operations The values stored within these registers do not affect the writes to the memory system. When a data fix event occurs, they are selected over the read data from ROM. If any part of the 1-word data fix is read, the entire word is replaced. So, a byte or half-word read causes the ROMC to replace the entire word. The word is word-address aligned.	0

45.4.21.2.2 ROMCP control register (ROMPATCHCNTL)

The ROMCP control register (ROMCP_ROMPATCHCNTL) contains the block disable bit and the data fix enable bits. The block disable bit provides a means to disable the ROMCP data fix and opcode patching functions, even when the address comparators are enabled. The External Boot feature is not affected by this bit. The eight data fix enable bits (0 through 7), when set, assign the associated address comparators to data fix operations. If more than one address comparator matches, the highest-numbered one takes precedence. The value in corresponding data register is used for the patching event.

Note: Bits 27 and 22 always read as 1s.

Table 765. ROMCP control register - Offset: F4h

Bit	Name	Access	Description and value	Reset value
31	LOCK	R/W	ROM lock This bit is sticky to the next reset. When asserted, the program model becomes RZWI except that the lock bit is RO. When the lock bit is set, a read of the ROMPATCHCNTL returns 0x8000_0000. All other valid ROMCP program model reads return 0x0000_0000 and any valid ROMCP program model write is ignored.	0
30	reserved	-	reserved	-
29	DIS	R/W	ROMCP disable; this bit, when set, disables all ROMCP operations This bit is used to enable secure operations.	0
			00b does not affect any ROMCP functions (default)	
			01b disables all ROMCP functions, data fixing, and opcode patching	
28:8	reserved	-	reserved	-
7:0	DATAFIX	R/W	data fix enable; controls the use of the first 8 address comparators for 1-word data fix or for code patch routine.	0
			0000_0000b address comparator triggers a opcode patch	
			0000_0001b address comparator triggers a data fix	

45.4.21.2.3 ROMCP enable register high (ROMPATCHENH)

The ROMCP enable register high (ROMCP_ROMPATCHENH) and ROMCP enable register low (ROMCP_ROMPATCHENL) control if the associated address comparator can trigger a opcode patch or data fix event. This implementation of the ROMCP only has 16 comparators. So, ROMCP_ROMPATCHENH and the upper half of ROMC_ROMPATCHENL are read-only. ROMCP_ROMPATCHENL[15:0] are associated with comparators 15 through 0. ROMC_ROMPATCHENLH[31:0] would have been associated with comparators 63 through 32.

Table 766. ROMCP enable register high - Offset: F8h

Bit	Name	Access	Description and value	Reset value
31:0	reserved	-	reserved	-

45.4.21.2.4 ROMC enable register low (ROMPATCHENL)

The ROMCP enable register high (ROMCP_ROMPATCHENH) and ROMCP enable register low (ROMCP_ROMPATCHENL) control if the associated address comparator can trigger a opcode patch or data fix event. This implementation of the ROMCP only has 16 comparators. So, ROMCP_ROMPATCHENH and the upper half of ROMC_ROMPATCHENL are read-only. ROMCP_ROMPATCHENL[15:0] are associated with comparators 15 through 0. ROMC_ROMPATCHENLH[31:0] would have been associated with comparators 63 through 32.

Table 767. ROMCP enable register low - Offset: FCh

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved	-
15:0	ENABLE	R/W	enable address comparator; this bit enables the corresponding address comparator to trigger an event 0000_0000_0000_0000b: address comparator disabled 0000_0000_0000_0001b: address comparator enabled, When the associated address is matched, ROMC triggers an opcode patch or data fix event.	0

45.4.21.2.5 ROMCP address registers (ROMPATCHaA)

The ROMCP address registers (ROMCP_ROMPATCHA0 through ROMCP_ROMPATCHA15) store the memory addresses where opcode patching begins and data fixing occurs. The address registers ROMCP_ROMPATCHA0 through ROMCP_ROMPATCHA15 are each 21 bits wide and dedicated to one 4 Mbyte memory space. Bits 21 through 2 are address bits to be compared with romc_haddr[21:2] for a match; bit 1 is also an address bit used for half-word selection. Bit 0 is the mode bit (set to 1 for THUMB mode). 1-word data fixing can only be used on the first 8 of the address comparators. ROMCP_ROMPATCHA0 through ROMCP_ROMPATCHA15 are each associated with address comparators 0 through 15.

For a = 0 to 15.

Table 768. ROMCP address registers - Offset: 100h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved	-
22:1	ADDRX	R/W	address comparator registers; indicates the memory address to be watched All 16 registers can be used for code patch address comparison. Only the first 8 registers can be used for a 1-word data fix address comparison. If data fix, bit 1 is ignored. Only used in code patch.	0
0	THUMBX	R/W	THUMB comparator select; indicates that this address triggers a THUMB opcode patch or an Arm opcode patch. If this watchpoint is selected to be a data fix, this bit is ignored as all data fixes are 1-word data fixes.	0
			0b Arm patch	
			1b THUMB patch (if data fix, ignore)	

45.4.21.2.6 ROMCP status register (ROMPATCHSR)

The ROMCP status register (ROMCP_ROMPATCHSR) indicates the current state of the ROMCP and the source number of the most recent address comparator event.

Table 769. ROMCP status registers - Offset: 208h

Bit	Name	Access	Description and value	Reset value
31:18	reserved	-	reserved	-
17	SW	R/W	ROMCP AHB multiple address comparator matches Indicator; indicates that multiple address comparator matches occurred. Writing a 1 to this bit clears it.	0
			0b no event or comparator collisions	
			1b a collision has occurred	
16:6	reserved	-	reserved	-
5:0	SOURCE	R	ROMCP source number Binary encoding of the number of the address comparator which has an address match in the most recent patch event on ROMCP AHB. If multiple matches occurred, the highest priority source number is used.	0
			00_0000b address comparator 0 matched	
			00_0001b address comparator 1 matched	
			00_0010b address comparator 2 matched	
			00_0011b address comparator 3 matched	
			00_0100b address comparator 4 matched	
			00_0101b address comparator 5 matched	
			00_0110b address comparator 6 matched	
			00_0111b address comparator 7 matched	
			00_1000b address comparator 8 matched	
			00_1001b address comparator 9 matched	
			00_1010b address comparator 10 matched	
			00_1011b address comparator 11 matched	

Table 769. ROMCP status registers - Offset: 208h...continued

Bit	Name	Access	Description and value		Reset value
			00_1100b	address comparator 12 matched	
			00_1101b	address comparator 13 matched	
			00_1110b	address comparator 14 matched	
			00_1111b	address comparator 15 matched	

45.5 AHB peripherals

[Table 770](#) provides details of the addresses for AHB peripherals. AHB peripherals have both secure and non-secure access.

Table 770. AHB peripherals memory map

AHB Port	Non-secure base address	Secure base Address	Peripheral	Reference
7	4008_2000h	5008_2000h	DMA engine (DMA0)	Section 45.5.1
	4008_5000h	5008_5000h	State configurable timer / PWM (SCT)	Section 45.5.2
	4008_6000h	5008_6000h	Flexcomm interface 0 (FC0)	Section 45.5.3
	4008_7000h	5008_7000h	Flexcomm interface 1 (FC1)	Section 45.5.3
	4008_8000h	5008_8000h	Flexcomm interface 2 (FC2)	Section 45.5.3
	4008_C000h	5008_C000h	High-speed GPIO (GPIO)	Section 45.5.7
8	4009_5000h	5009_5000h	CRC engine (CRC)	Section 45.5.8
	4009_C000h	5009_C000h	Debug mailbox (DBG)	Section 45.5.9
	4009_F000h	5009_F000h	SPIFI slave port (SPIFI)	Section 45.5.10
9	400A_1000h	500A_1000h	Code watchdog (CDOG)	Section 45.5.11
	400A_4000h	500A_4000h	Hash-AES peripheral (HASHAES)	Section 45.5.12
	400A_5000h	500A_5000h	Crypto coprocessor (CASPER)	Section 45.5.13
	400A_C000h	500A_C000h	AHB security control (AHB_SECURE_CTRL)	Section 45.5.14
10	400B_0000h	500B_0000h	Bluetooth Low Energy radio (BLE)	Section 44

45.5.1 DMA engine (DMA0)

45.5.1.1 DMA memory map

The DMA0 registers are located at base address: 4008_2000h

The DMA1 registers are located at base address: 400A_7000h

Table 771. DMA engine registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CTRL	R/W	0h	DMA control	32		Section 45.5.1.2.1
INTSTAT	R	4h	interrupt status	32		Section 45.5.1.2.2
SRAMBASE	R/W	8h	SRAM address of the channel configuration table	32		Section 45.5.1.2.3
ENABLESET0	R/W	20h	channel enable read and set for all DMA channels	32	0000_0000h	Section 45.5.1.2.4
ENABLECLR0	W	28h	channel enable clear for all DMA channels	32		Section 45.5.1.2.5
ACTIVE0	R	30h	channel active status for all DMA channels	32	0000_0000h	Section 45.5.1.2.6
BUSY0	R	38h	channel busy status for all DMA channels	32	0000_0000h	Section 45.5.1.2.7
ERRINT0	R/W	40h	error interrupt status for all DMA channels	32	0000_0000h	Section 45.5.1.2.8
INTENSET0	R/W	48h	interrupt enable read and Set for all DMA channels	32	0000_0000h	Section 45.5.1.2.9
INTENCLR0	W	50h	interrupt enable clear for all DMA channels	32		Section 45.5.1.2.10
INTA0	R/W	68h	interrupt A status for all DMA channels	32	0000_0000h	Section 45.5.1.2.11
INTB0	R/W	60h	interrupt B status for all DMA channels	32	0000_0000h	Section 45.5.1.2.12
SETVALID0	W	68h	set ValidPending control bits for all DMA channels	32		Section 45.5.1.2.13
SETTRIG0	W	70h	Set trigger control bits for all DMA channels	32		Section 45.5.1.2.14
ABORT0	W	78h	channel abort control for all DMA channels	32		Section 45.5.1.2.15
CFG0 - CFG22	R/W	400h - 560h	configuration register for DMA channel n	32		Section 45.5.1.2.16
CTLSTAT0 - CTLSTAT22	R	404h - 564h	control and status register for DMA channel n	32		Section 45.5.1.2.17
XFERCFG0 - XFERCFG22	R/W	408h - 568h	transfer configuration register for DMA channel n	32		Section 45.5.1.2.18

45.5.1.2 Register descriptions

45.5.1.2.1 DMA control (CTRL)

Table 772. DMA control - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved Read value is undefined, only zero must be written.	-
0	ENABLE	R/W	DMA controller master enable 0b disabled The DMA controller is disabled. Clears any triggers that were asserted at the point when disabled, but does not prevent retriggering when the DMA controller is reenabled. 1b enabled The DMA controller is enabled	0

45.5.1.2.2 Interrupt status (INTSTAT)

Table 773. Interrupt status - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2	ACTIVEERRINT	R	summarizes if any error interrupts are pending 0b not pending No enabled interrupts are pending. 1b pending At least one error interrupt is pending.	0
1	ACTIVEINT	R	summarizes if any enabled interrupts (other than error interrupts) are pending 0b not pending No enabled interrupts are pending. 1b pending At least one error interrupt is pending.	0
0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.5.1.2.3 SRAM address of the channel configuration table (SRAMBASE)

Table 774. SRAM address of the channel configuration table - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:9	OFFSET	R/W	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
8:0	reserved	-	reserved Read value is undefined, only zero must be written.	0

45.5.1.2.4 Channel enable read and set for all DMA channels (ENABLESET0)

Table 775. Channel enable read and set for all DMA channels - Offset: 20h

Bit	Name	Access	Description and value	Reset value
31:0	ENA	R/W	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
			0 disabled	
			1 enabled	

45.5.1.2.5 Channel enable clear for all DMA channels (ENABLECLR0)

Table 776. Channel enable clear for all DMA channels - Offset: 28h

Bit	Name	Access	Description and value	Reset value
31:0	CLR	W	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.	0

45.5.1.2.6 Channel active status for all DMA channels (ACTIVE0)

Table 777. Channel active status for all DMA channels - Offset: 30h

Bit	Name	Access	Description and value	Reset value
31:0	ACT	R	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
			0 not active	
			1 active	

45.5.1.2.7 Channel busy status for all DMA channels (BUSY0)

Table 778. Channel busy status for all DMA channels - Offset: 38h

Bit	Name	Access	Description and value	Reset value
31:0	BSY	R	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
			0 not busy	
			1 busy	

45.5.1.2.8 Error interrupt status for all DMA channels (ERRINT0)

Table 779. Error interrupt status for all DMA channels - Offset: 40h

Bit	Name	Access	Description and value	Reset value
31:0	ERR	R/W	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
			0 error interrupt is not active	
			1 error interrupt is active	

45.5.1.2.9 Interrupt enable read and set for all DMA channels (INTENSET0)

Table 780. Interrupt enable read and set for all DMA channels - Offset: 48h

Bit	Name	Access	Description and value	Reset value
31:0	INTEN	R/W	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
			0 interrupt for DMA channel is disabled	
			1 interrupt for DMA channel is enabled	

45.5.1.2.10 Interrupt enable clear for all DMA channels (INTENCLR0)

Table 781. Interrupt enable clear for all DMA channels - Offset: 50h

Bit	Name	Access	Description and value	Reset value
31:0	CLR	W	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.	0

45.5.1.2.11 Interrupt A status for all DMA channels (INTA0)

Table 782. Interrupt A status for all DMA channels - Offset: 58h

Bit	Name	Access	Description and value	Reset value
31:0	IA	R/W	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
			0 DMA channel interrupt A is not active	
			1 DMA channel interrupt A is active	

45.5.1.2.12 Interrupt B status for all DMA channels (INTB0)

Table 783. Interrupt B status for all DMA channels - Offset: 60h

Bit	Name	Access	Description and value	Reset value
31:0	IB	R/W	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
			0 DMA channel interrupt B is not active	
			1 DMA channel interrupt B is active	

45.5.1.2.13 Set ValidPending control bits for all DMA channels (SETVALID0)

Table 784. Set ValidPending control bits for all DMA channels - Offset: 68h

Bit	Name	Access	Description and value	Reset value
31:0	SV	R/W	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
			0 no effect	
			1 sets the VALIDPENDING control bit for DMA channel n	

45.5.1.2.14 Set trigger control bits for all DMA channels (SETTRIG0)

Table 785. Set trigger control bits for all DMA channels - Offset: 70h

Bit	Name	Access	Description and value	Reset value
31:0	TRIG	W	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
			0 no effect	
			1 sets the TRIG bit for DMA channel n.	

45.5.1.2.15 Channel abort control for all DMA channels (ABORT0)

Table 786. Channel abort control for all DMA channels - Offset: 78h

Bit	Name	Access	Description and value	Reset value
31:0	ABORTCTRL	W	abort control for DMA channel 0 Bit n corresponds to DMA channel n.	0
			0 no effect	
			1 aborts DMA operations on channel n	

45.5.1.2.16 Configuration register for DMA channel n (CFGn)

For n = 0 to 22

Note: Each module instance supports a different number of registers.

Table 787. Register reset values

Register	Reset value
CFG0 to CFG9	DMA0; DMA1: 0000_0000h
CFG10 to CFG22	DMA0: 0000_0000h
	DMA1: Register not supported

Table 788. Configuration register for DMA channel n - Offset: 400h + (n × 10h)

Bit	Name	Access	Description and value	Reset value
31:19	reserved	-	reserved Read value is undefined, only zero must be written.	-
18:16	CHPRIORITY	R/W	priority of this channel when multiple DMA requests are pending Eight priority levels are supported: 0x0 = highest priority, 0x7 = lowest priority	See Table 787
15	DSTBURSTWRAP	R/W	destination burst wrap When enabled, the destination data address for the DMA is "wrapped", meaning that the destination address range for each burst is the same. As an example, it can be used to write several sequential registers to a peripheral for each DMA burst, writing the same registers again for each burst.	See Table 787
			0b disabled Destination burst wrapping is not enabled for this DMA channel.	
			1b enabled Destination burst wrapping is enabled for this DMA channel.	

Table 788. Configuration register for DMA channel n - Offset: 400h + (n × 10h)...continued

Bit	Name	Access	Description and value		Reset value										
14	SRCBURSTWRAP	R/W	source burst wrap When enabled, the source data address for the DMA is "wrapped", meaning that the source address range for each burst is the same. As an example, it can be used to read several sequential registers from a peripheral for each DMA burst, reading the same registers again for each burst. <table><tr><td>0b</td><td>disabled</td></tr><tr><td colspan="2">Source burst wrapping is not enabled for this DMA channel.</td></tr><tr><td>1b</td><td>enabled</td></tr><tr><td colspan="2">Source burst wrapping is enabled for this DMA channel.</td></tr></table>		0b	disabled	Source burst wrapping is not enabled for this DMA channel.		1b	enabled	Source burst wrapping is enabled for this DMA channel.		See Table 787		
0b	disabled														
Source burst wrapping is not enabled for this DMA channel.															
1b	enabled														
Source burst wrapping is enabled for this DMA channel.															
13:12	reserved	-	reserved Read value is undefined, only zero must be written.		-										
11:8	BURSTPOWER	R/W	burst power Burst Power is used in two ways. When SRCBURSTWRAP and/or DSTBURSTWRAP modes are selected, it always selects the address wrap size (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 feature can be used, for example, with peripherals that contain a FIFO that can initiate a DMA operation when the FIFO reaches a certain level. The values below corresponds to the maximum supported transfer count. The total transfer length as defined in the XFERCOUNT bits in the XFERCFG register must be an even multiple of the burst size. <table><tr><td>0000</td><td>burst size = 1 (20)</td></tr><tr><td>0001</td><td>burst size = 2 (21)</td></tr><tr><td>0010</td><td>burst size = 4 (22)</td></tr><tr><td>1010</td><td>burst size = 1024 (210)</td></tr><tr><td>others</td><td>not supported</td></tr></table>		0000	burst size = 1 (20)	0001	burst size = 2 (21)	0010	burst size = 4 (22)	1010	burst size = 1024 (210)	others	not supported	See Table 787
0000	burst size = 1 (20)														
0001	burst size = 2 (21)														
0010	burst size = 4 (22)														
1010	burst size = 1024 (210)														
others	not supported														
7	reserved	-	reserved Read value is undefined, only zero must be written.		-										

Table 788. Configuration register for DMA channel n - Offset: 400h + (n × 10h)...continued

Bit	Name	Access	Description and value	Reset value
6	TRIGBURST	R/W	trigger burst Selects if hardware triggers cause a single or burst transfer.	See Table 787
			0b single transfer Hardware trigger causes a single transfer.	
			1b 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.	
5	TRIGTYPE	R/W	trigger type Selects hardware trigger as edge triggered or level triggered.	See Table 787
			0b edge Hardware trigger is edge triggered. Transfers are initiated and completed, as specified for a single trigger.	
			1b level Hardware trigger is level triggered. When level triggering without burst (BURSTPOWER = 0) is selected, only hardware triggers must be used on that channel. Transfers continue as long as the trigger level is asserted. When the trigger is deasserted, the transfer is paused until the trigger is asserted again. However, the transfer is not paused until all remaining transfers within the current BURSTPOWER length are completed.	
4	TRIGPOL	R/W	trigger polarity Selects the polarity of a hardware trigger for this channel.	See Table 787
			0b active low - falling edge Hardware trigger is active low or falling edge triggered, based on TRIGTYPE.	
			1b active high - rising edge Hardware trigger is active high or rising edge triggered, based on TRIGTYPE.	

Table 788. Configuration register for DMA channel n - Offset: 400h + (n × 10h)...continued

Bit	Name	Access	Description and value	Reset value
3:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	HWTRIGEN	R/W	hardware triggering enable for this channel	See Table 787
			0b disabled Hardware triggering is not used.	
			1b enabled Hardware triggering is used.	
0	PERIPHREQEN	R/W	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.	See Table 787
			0b disabled Peripheral DMA requests are disabled.	
			1b enabled Peripheral DMA requests are enabled.	

45.5.1.2.17 Control and status register for DMA channel n (CTLSTATn)

For n = 0 to 22

Note: Each module instance supports a different number of registers.

Table 789. Register reset values

Instance	Register supported	Register not supported
DMA0	CTLSTAT0 to CTLSTAT22	-
DMA1	CTLSTAT0 to CTLSTAT9	CTLSTAT10 to CTLSTAT22

Table 790. Control and status register for DMA channel n - Offset: 404h + (n × 10h)

Bit	Name	Access	Description and value		Reset value				
31:3	reserved	-	reserved Read value is undefined, only zero must be written.		-				
2	TRIG	R	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. <table><tr><td>0b</td><td>not triggered The trigger for this DMA channel is not set. DMA operations is not carried out.</td></tr><tr><td>1b</td><td>triggered The trigger for this DMA channel is set. DMA operations is carried out.</td></tr></table>		0b	not triggered The trigger for this DMA channel is not set. DMA operations is not carried out.	1b	triggered The trigger for this DMA channel is set. DMA operations is carried out.	See Table 789
0b	not triggered The trigger for this DMA channel is not set. DMA operations is not carried out.								
1b	triggered The trigger for this DMA channel is set. DMA operations is carried out.								
1	reserved	-	reserved Read value is undefined, only zero must be written.		-				
0	VALIDPENDING	R/W	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. <table><tr><td>0b</td><td>no effect No effect on DMA operation.</td></tr><tr><td>1b</td><td>valid pending</td></tr></table>		0b	no effect No effect on DMA operation.	1b	valid pending	See Table 789
0b	no effect No effect on DMA operation.								
1b	valid pending								

45.5.1.2.18 Transfer configuration register for DMA channel n (XFERCFGn)

For n = 0 to 22

Note: Each module instance supports a different number of registers.

Table 791. Register reset values

Instance	Register supported	Register not supported
DMA0	XFERCFG0 to XFERCFG22	-
DMA1	XFERCFG0 to XFERCFG9	XFERCFG10 to XFERCFG22

Table 792. Transfer configuration register for DMA channel n - Offset: 408h + (n × 10h)

Bit	Name	Access	Description and value	Reset value
31:26	reserved	-	reserved Read value is undefined, only zero must be written.	-

Table 792. Transfer configuration register for DMA channel n - Offset: 408h + (n × 10h)...continued

Bit	Name	Access	Description and value	Reset value
25:16	XFERCOUNT	R/W	Total number of transfers to be performed, minus 1 encoded The number of bytes transferred is: (XFERCOUNT + 1) × data width (as defined by the WIDTH field). The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 = 1 transfer is performed; 0x1 = 2 transfers are performed. 0x3FF = 1,024 transfers are performed.	See Table 791
15:14	DSTINC	R/W	determines whether the destination address is incremented for each DMA transfer 00b no increment The destination address is not incremented for each transfer. Usual case when the destination is a peripheral device. 01b 1 × width The destination address is incremented by the number specified by width for each transfer. Usual case when the destination is memory. 10b 2 × width The destination address is incremented by 2 times the number specified by width for each transfer. 11b 4 × width The destination address is incremented by 4 times the number specified by width for each transfer.	See Table 791
13:12	SRCINC	R/W	determines if the source address is incremented for each DMA transfer 00b no increment The source address is not incremented for each transfer. Usual case when the source is a peripheral device. 01b 1 × width The source address is incremented by the number specified by width for each transfer. Usual case when the source is memory. 10b 2 × width The source address is incremented by 2 times the number specified by width for each transfer. 11b 4 × width The source address is incremented by 4 times the number specified by width for each transfer.	See Table 791
11:10	reserved	-	reserved Read value is undefined, only zero must be written.	-

Table 792. Transfer configuration register for DMA channel n - Offset: 408h + (n × 10h)...continued

Bit	Name	Access	Description and value		Reset value
9:8	WIDTH	R/W	transfer width used for this DMA channel		See Table 791
			00b	8-bit 8-bit transfers are performed (8-bit source reads and destination writes).	
			01b	16-bit 16-bit transfers are performed (16-bit source reads and destination writes).	
			10b	32-bit 32-bit transfers are performed (32-bit source reads and destination writes).	
			11b	reserved Reserved setting. Do not use.	
7:6	reserved	-	reserved Read value is undefined, only zero must be written.		-
5	SETINTB	R/W	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 required.		See Table 791
			0b	no effect	
			1b	set When the current descriptor is exhausted, the INTB flag for this channel is set.	
4	SETINTA	R/W	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 required.		See Table 791
			0b	no effect	
			1b	set When the current descriptor is exhausted, the INTA flag for this channel is set.	
3	CLRTRIG	R/W	clear trigger		See Table 791
			0b	not cleared The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor is started.	
			1b	cleared The trigger is cleared when this descriptor is exhausted.	

Table 792. Transfer configuration register for DMA channel n - Offset: 408h + (n × 10h)...continued

Bit	Name	Access	Description and value		Reset value
2	SWTRIG	R/W	software trigger		See Table 791
			0b	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 is required to start the channel.	
			1b	set When written by software, the trigger for this channel is set immediately. This feature must not be used with level triggering when TRIGBURST = 0.	
1	RELOAD	R/W	indicates if the control structure of the channel is reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers.		See Table 791
			0b	disabled Do not reload the control structure of the channels when the current descriptor is exhausted.	
			1b	enabled Reload the control structure of the channels when the current descriptor is exhausted.	
0	CFGVALID	R/W	configuration valid flag This bit indicates if the current channel descriptor is valid and can potentially be acted upon if all other activation criteria are fulfilled.		See Table 791
			0b	not valid The channel descriptor is not considered valid until validated by an associated SETVALID0 setting.	
			1b	valid The current channel descriptor is considered valid.	

45.5.2 State configurable timer/PWM (SCT)

45.5.2.1 SCT memory map

The SCT registers are located at base address: 4008_5000h.

Table 793. State configurable timer/pulse width modulator registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CONFIG	R/W	0h	SCT configuration register	32		Section 45.5.2.2.1
CTRL	R/W	4h	SCT control register	32		Section 45.5.2.2.2
LIMIT	R/W	8h	SCT limit event select register	32	0000_0000h	Section 45.5.2.2.3
HALT	R/W	Ch	SCT halt event select register	32	0000_0000h	Section 45.5.2.2.4
STOP	R/W	10h	SCT stop event select register	32	0000_0000h	Section 45.5.2.2.5
START	R/W	14h	SCT start event select register	32	0000_0000h	Section 45.5.2.2.6
COUNT	R/W	40h	SCT counter register	32	0000_0000h	Section 45.5.2.2.7
STATE	R/W	44h	SCT state register	32		Section 45.5.2.2.8
INPUT	R	48h	SCT input register	32	0000_0000h	Section 45.5.2.2.9
REGMODE	R/W	4Ch	SCT match/capture mode register	32	0000_0000h	Section 45.5.2.2.10
OUTPUT	R/W	50h	SCT output register	32		Section 45.5.2.2.11
OUTPUTDIRCTRL	R/W	54h	SCT output counter direction control register	32	0000_0000h	Section 45.5.2.2.12
RES	R/W	58h	SCT conflict resolution register	32	0000_0000h	Section 45.5.2.2.13
DMAREQ0	R/W	5Ch	SCT DMA request 0 register	32		Section 45.5.2.2.14
DMAREQ1	R/W	60h	SCT DMA request 1 register	32		Section 45.5.2.2.15
EVEN	R/W	F0h	SCT event interrupt enable register	32		Section 45.5.2.2.16
EVFLAG	R/W	F4h	SCT event flag register	32		Section 45.5.2.2.17
CONEN	R/W	F8h	SCT conflict interrupt enable register	32		Section 45.5.2.2.18
CONFLAG	R/W	FCh	SCT conflict flag register	32		Section 45.5.2.2.19
CAP0 - CAP15	R/W	100h - 13Ch	SCT capture register of capture channel	32	0000_0000h	Section 45.5.2.2.20
MATCH0 - MATCH15	R/W	100h - 13Ch	SCT match value register of match channels	32	0000_0000h	Section 45.5.2.2.21
CAPCTRL0 - CAPCTRL15	R/W	200h - 23Ch	SCT capture control register	32	0000_0000h	Section 45.5.2.2.22

Table 793. State configurable timer/pulse width modulator registers...continued

Name	Access	Offset	Description	Width (bits)	Reset value	Section
MATCHREL0 - MATCHREL15	R/W	200h - 23Ch	SCT match reload value register	32	0000_0000h	Section 45.5.2.2.23
EV0_STATE - EV15_STATE	R/W	300h - 378h	SCT event state register n	32		Section 45.5.2.2.24
EV0_CTRL - EV15_CTRL	R/W	304h - 37Ch	SCT event control register n	32		Section 45.5.2.2.25
OUT0_SET - OUT9_SET	R/W	500h - 548h	SCT output n set register	32		Section 45.5.2.2.26
OUT0_CLR - OUT9_CLR	R/W	504h - 54Ch	SCT output n clear register	32		Section 45.5.2.2.27

45.5.2.2 Register descriptions

45.5.2.2.1 SCT configuration register (CONFIG)

Table 794. SCT configuration register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:19	reserved	-	reserved	-
18	AUTOLIMIT_H	R/W	automatic limit high A one in this bit causes a match on match register 0 to be treated as a de-facto LIMIT condition without having to define an associated event. 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.	0
17	AUTOLIMIT_L	R/W	automatic limit low A one in this bit causes a match on match register 0 to be treated as a de-facto LIMIT condition without having to define an associated event. 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.	0
16:13	reserved	-	reserved	-
12:9	INSYNC	R/W	Synchronization for input N (bit 9 = input 0, bit 10 = input 1, bit 12 = input 3); all other bits are reserved A 1 in one of these bits subjects the corresponding input to synchronization to the SCT clock, before it is used to create an event. If an input is known to be synchronous to the SCT clock already, this bit may be set to 0 for faster input response. Note: The SCT clock is the system clock for CKMODEs 0-2. It is the selected, asynchronous SCT input clock for CKMODE3 Note: The INSYNC field only affects inputs used for event generation. It does not apply to the clock input specified in the CKSEL field.	1

Table 794. SCT configuration register - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value	
8	NORELOAD_H	R/W	no reload high A 1 in this bit prevents the higher match registers from being reloaded from their respective reload registers. Setting this bit eliminates having 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	
7	NORELOAD_L	R/W	no reload low A 1 in this bit prevents the lower match registers from being reloaded from their respective reload registers. Setting this bit eliminates having 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.		
6:3	CKSEL	R/W	SCT clock select The specific functionality of the designated input/edge depends on the CLKMODE bit selection in this register.	0	
			0000b		rising edges on input 0
			0001b		falling edges on input 0
			0010b		rising edges on input 1
			0011b		falling edges on input 1
			0100b		rising edges on input 2
			0101b		falling edges on input 2
			0110b		rising edges on input 3
			0111b		falling edges on input 3
			1000b		rising edges on input 4
			1001b		falling edges on input 4
			1010b		rising edges on input 5
			1011b		falling edges on input 5
			1100b		rising edges on input 6
			1101b		falling edges on input 6
			1110b		rising edges on input 7
1111b	falling edges on input 7				

Table 794. SCT configuration register - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value
2:1	CLKMODE	R/W	SCT clock mode	0
			00b system clock mode The system clock clocks the entire SCT module including the counter(s) and counter prescalers.	
			01b 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.	
			10b 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 pulse width on the clock input is 1 bus clock period. This mode is the low-power, sampled-clock mode.	
			11b 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 - 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.	
0	UNIFY	R/W	SCT operation	0
			0b The SCT operates as two 16-bit counters named COUNTER_L and COUNTER_H.	
			1b The SCT operates as a unified 32-bit counter.	

45.5.2.2.2 SCT control register (CTRL)

Table 795. SCT control register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:29	reserved	-	reserved	-
28:21	PRE_H	R/W	prescale high 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. Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.	0
20	BIDIR_H	R/W	direction select	0
			0b The H counter counts up to its limit condition, then is cleared to zero.	
			1b The H counter counts up to its limit, then counts down to a limit condition or to 0.	

Table 795. SCT control register - Offset: 4h...continued

Bit	Name	Access	Description and value	Reset value
19	CLRCTRL_H	R/W	clear control high Writing a 1 to this bit clears the H counter. This bit always reads as 0.	-
18	HALT_H	R/W	halt high 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. When set, this bit can only be cleared by software to restore counter operation. This bit is set on reset.	1
17	STOP_H	R/W	stop high When this bit is 1 and HALT is 0, the H counter does not run. However, 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
16	DOWN_H	R/W	count down high This 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 = 1. Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0.	0
15:13	reserved	-	reserved	-
12:5	PRE_L	R/W	prescale low 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. Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.	0
4	BIDIR_L	R/W	L or unified counter direction select	0
			0b up The counter counts up to a limit condition, then is cleared to zero.	
			1b up-down The counter counts up to a limit, then counts down to a limit condition or to 0.	
3	CLRCTRL_L	R/W	clear control low Writing a 1 to this bit clears the L or unified counter. This bit always reads as 0.	0
2	HALT_L	R/W	halt low 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. When set, only software can clear this bit to restore counter operation. This bit is set on reset.	1

Table 795. SCT control register - Offset: 4h...continued

Bit	Name	Access	Description and value	Reset value
1	STOP_L	R/W	stop low When this bit is 1 and HALT is 0, the L or unified counter does not run. However, I/O events related to the counter can occur. If a designated start event occurs, this bit is cleared and counting resumes.	0
0	DOW_L	R/W	down count low This 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

45.5.2.2.3 SCT limit event select register (LIMIT)

Table 796. SCT limit event select register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:16	LIMMSK_H	R/W	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, and so on.). The number of bits = the number of events in this SCT.	0
15:0	LIMMSK_L	R/W	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, and so on). The number of bits = the number of events in this SCT.	0

45.5.2.2.4 SCT halt event select register (HALT)

Table 797. SCT halt event select register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:16	HALTMSK_H	R/W	If bit n is one, event n sets the HALT_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, and so on). The number of bits = the number of events in this SCT.	0
15:0	HALTMSK_L	R/W	If bit n is one, event n sets the HALT_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, and so on). The number of bits = the number of events in this SCT.	0

45.5.2.2.5 SCT stop event select register (STOP)

Table 798. SCT stop event select register - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:16	STOPMSK_H	R/W	If bit n is one, event n sets the STOP_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, and so on). The number of bits = the number of events in this SCT.	0
15:0	STOPMSK_L	R/W	If bit n is one, event n sets the STOP_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, and so on). The number of bits = the number of events in this SCT.	0

45.5.2.2.6 SCT start event select register (START)

Table 799. SCT start event select register - Offset: 14h

Bit	Name	Access	Description and value	Reset value
31:16	STARTMSK_H	R/W	If bit n is one, event n sets the START_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, and so on). The number of bits = the number of events in this SCT.	0
15:0	STARTMSK_L	R/W	If bit n is one, event n sets the START_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, and so on). The number of bits = the number of events in this SCT.	0

45.5.2.2.7 SCT counter register (COUNT)

Table 800. SCT counter register - Offset: 40h

Bit	Name	Access	Description and value	Reset value
31:16	CTR_H	R/W	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
15:0	CTR_L	R/W	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

45.5.2.2.8 SCT state register (STATE)

Table 801. SCT state register - Offset: 44h

Bit	Name	Access	Description and value	Reset value
31:21	reserved	-	reserved	-
20:16	STATE_H	R/W	state variable	0
15:5	reserved	-	reserved	-
4:0	STATE_L	R/W	state variable	0

45.5.2.2.9 SCT input register (INPUT)

Table 802. SCT input register - Offset: 48h

Bit	Name	Access	Description and value	Reset value
31	SIN15	R	input 15 state Input 15 state following the synchronization specified by INSYNCR.	0
30	SIN14	R	input 14 state Input 14 state following the synchronization specified by INSYNCR.	0
29	SIN13	R	input 13 state Input 13 state following the synchronization specified by INSYNCR.	0
28	SIN12	R	input 12 state Input 12 state following the synchronization specified by INSYNCR.	0

Table 802. SCT input register - Offset: 48h...continued

Bit	Name	Access	Description and value	Reset value
27	SIN11	R	input 11 state Input 11 state following the synchronization specified by INSYNC.	0
26	SIN10	R	input 10 state Input 10 state following the synchronization specified by INSYNC.	0
25	SIN9	R	input 9 state Input 9 state following the synchronization specified by INSYNC.	0
24	SIN8	R	input 8 state Input 8 state following the synchronization specified by INSYNC.	0
23	SIN7	R	input 7 state Input 7 state following the synchronization specified by INSYNC.	0
22	SIN6	R	input 6 state Input 6 state following the synchronization specified by INSYNC.	0
21	SIN5	R	input 5 state Input 5 state following the synchronization specified by INSYNC.	0
20	SIN4	R	input 4 state Input 4 state following the synchronization specified by INSYNC.	0
19	SIN3	R	input 3 state Input 3 state following the synchronization specified by INSYNC.	0
18	SIN2	R	input 2 state Input 2 state following the synchronization specified by INSYNC.	0
17	SIN1	R	input 1 state Input 1 state following the synchronization specified by INSYNC.	0
16	SIN0	R	input 0 state Input 0 state following the synchronization specified by INSYNC.	0
15	AIN15	R	input 15 state Input 15 state on the last SCT clock edge.	0
14	AIN14	R	input 14 state Input 14 state on the last SCT clock edge.	0
13	AIN13	R	input 13 state Input 13 state on the last SCT clock edge.	0
12	AIN12	R	input 12 state Input 12 state on the last SCT clock edge.	0
11	AIN11	R	input 11 state Input 11 state on the last SCT clock edge.	0
10	AIN10	R	input 10 state Input 10 state on the last SCT clock edge.	0
9	AIN9	R	input 9 state Input 9 state on the last SCT clock edge.	0
8	AIN8	R	input 8 state Input 8 state on the last SCT clock edge.	0

Table 802. SCT input register - Offset: 48h...continued

Bit	Name	Access	Description and value	Reset value
7	AIN7	R	input 7 state Input 7 state on the last SCT clock edge.	0
6	AIN6	R	input 6 state Input 6 state on the last SCT clock edge.	0
5	AIN5	R	input 5 state Input 5 state on the last SCT clock edge.	0
4	AIN4	R	input 4 state Input 4 state on the last SCT clock edge.	0
3	AIN3	R	input 3 state Input 3 state on the last SCT clock edge.	0
2	AIN2	R	input 2 state Input 2 state on the last SCT clock edge.	0
1	AIN1	R	input 1 state Input 1 state on the last SCT clock edge.	0
0	AIN0	R	input 0 state Input 0 state on the last SCT clock edge.	0

45.5.2.2.10 SCT match/capture mode register (REGMODE)

Table 803. SCT match/capture mode register - Offset: 4Ch

Bit	Name	Access	Description and value	Reset value
31:16	REGMODE_H	R/W	register mode high Each bit controls one match/capture register (register 0 = bit 16, register 1 = bit 17, and so on). The number of bits = the number of match/captures in this SCT.	0
			0 register operates as match registers	
			1 register operates as capture registers	
15:0	REGMODE_L	R/W	register mode low Each bit controls one match/capture register (register 0 = bit 0, register 1 = bit 1, and so on). The number of bits = the number of match/captures in this SCT. 0 = register operates as match register. 1 = register operates as capture register.	0
			0 register operates as match registers	
			1 register operates as capture registers	

45.5.2.2.11 SCT output register (OUTPUT)

Table 804. SCT output register - Offset: 50h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved	-
15:0	OUT	R/W	output 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, and so on). The number of bits = the number of outputs in this SCT.	0

45.5.2.2.12 SCT output counter direction control register (OUTPUTDIRCTRL)

Table 805. SCT output counter direction control register - Offset: 54h

Bit	Name	Access	Description and value	Reset value
31:30	SETCLR15	R/W	set/clear operation on output 15 Value 0x3 is reserved. Do not program this value.	0
29:28	SETCLR14	R/W	set/clear operation on output 14 Value 0x3 is reserved. Do not program this value.	0
27:26	SETCLR13	R/W	set/clear operation on output 13 Value 0x3 is reserved. Do not program this value.	0
25:24	SETCLR12	R/W	set/clear operation on output 12 Value 0x3 is reserved. Do not program this value.	0
23:22	SETCLR11	R/W	set/clear operation on output 11 Value 0x3 is reserved. Do not program this value.	0
21:20	SETCLR10	R/W	set/clear operation on output 10 Value 0x3 is reserved. Do not program this value.	0
19:18	SETCLR9	R/W	set/clear operation on output 9 Value 0x3 is reserved. Do not program this value.	0
17:16	SETCLR8	R/W	set/clear operation on output 8 Value 0x3 is reserved. Do not program this value.	0
15:14	SETCLR7	R/W	set/clear operation on output 7 Value 0x3 is reserved. Do not program this value.	0
13:12	SETCLR6	R/W	set/clear operation on output 6 Value 0x3 is reserved. Do not program this value.	0
11:10	SETCLR5	R/W	set/clear operation on output 5 Value 0x3 is reserved. Do not program this value.	0
9:8	SETCLR4	R/W	set/clear operation on output 4 Value 0x3 is reserved. Do not program this value.	0
7:6	SETCLR3	R/W	set/clear operation on output 3 Value 0x3 is reserved. Do not program this value.	0
5:4	SETCLR2	R/W	set/clear operation on output 2 Value 0x3 is reserved. Do not program this value.	0

Table 805. SCT output counter direction control register - Offset: 54h...continued

Bit	Name	Access	Description and value	Reset value
3:2	SETCLR1	R/W	set/clear operation on output 1 Value 0x3 is reserved. Do not program this value.	0
1:0	SETCLR0	R/W	set/clear operation on output 0 Value 0x3 is reserved. Do not program this value.	0

45.5.2.2.13 SCT conflict resolution register (RES)

Table 806. SCT conflict resolution register - Offset: 58h

Bit	Name	Access	Description and value	Reset value
31:30	O15RES	R/W	effect of simultaneous set and clear on output 15	0
			00b no change	
			01b set output (or clear based on the SETCLR15 field in the OUTPUTDIRCTRL register)	
			10b clear output (or set based on the SETCLR15 field)	
			11b toggle output	
29:28	O14RES	R/W	effect of simultaneous set and clear on output 14	0
			00b no change	
			01b set output (or clear based on the SETCLR14 field in the OUTPUTDIRCTRL register)	
			10b clear output (or set based on the SETCLR14 field)	
			11b toggle output	
27:26	O13RES	R/W	effect of simultaneous set and clear on output 13	0
			00b no change	
			01b set output (or clear based on the SETCLR13 field in the OUTPUTDIRCTRL register)	
			10b clear output (or set based on the SETCLR13 field)	
			11b toggle output	
25:24	O12RES	R/W	effect of simultaneous set and clear on output 12	0
			00b no change	
			01b set output (or clear based on the SETCLR12 field in the OUTPUTDIRCTRL register)	
			10b clear output (or set based on the SETCLR12 field)	
			11b toggle output	

Table 806. SCT conflict resolution register - Offset: 58h...continued

Bit	Name	Access	Description and value		Reset value
23:22	O11RES	R/W	effect of simultaneous set and clear on output 11		0
			00b	no change	
			01b	set output (or clear based on the SETCLR11 field in the OUTPUTDIRCTRL register)	
			10b	clear output (or set based on the SETCLR11 field)	
			11b	toggle output	
21:20	O10RES	R/W	effect of simultaneous set and clear on output 10		0
			00b	no change	
			01b	set output (or clear based on the SETCLR10 field in the OUTPUTDIRCTRL register)	
			10b	clear output (or set based on the SETCLR10 field)	
			11b	toggle output	
19:18	O9RES	R/W	effect of simultaneous set and clear on output 9		0
			00b	no change	
			01b	set output (or clear based on the SETCLR9 field in the OUTPUTDIRCTRL register)	
			10b	clear output (or set based on the SETCLR9 field)	
			11b	toggle output	
17:16	O8RES	R/W	effect of simultaneous set and clear on output 8		0
			00b	no change	
			01b	set output (or clear based on the SETCLR8 field in the OUTPUTDIRCTRL register)	
			10b	clear output (or set based on the SETCLR8 field)	
			11b	toggle output	
15:14	O7RES	R/W	effect of simultaneous set and clear on output 7		0
			00b	no change	
			01b	set output (or clear based on the SETCLR7 field in the OUTPUTDIRCTRL register)	
			10b	clear output (or set based on the SETCLR7 field)	
			11b	toggle output	
13:12	O6RES	R/W	effect of simultaneous set and clear on output 6		0
			00b	no change	
			01b	set output (or clear based on the SETCLR6 field in the OUTPUTDIRCTRL register)	
			10b	set output (or clear based on the SETCLR6 field in the OUTPUTDIRCTRL register)	
			11b	toggle output	

Table 806. SCT conflict resolution register - Offset: 58h...continued

Bit	Name	Access	Description and value	Reset value
11:10	O5RES	R/W	effect of simultaneous set and clear on output 5	0
			00b no change	
			01b set output (or clear based on the SETCLR5 field in the OUTPUTDIRCTRL register)	
			10b set output (or clear based on the SETCLR5 field in the OUTPUTDIRCTRL register)	
			11b toggle output	
9:8	O4RES	R/W	effect of simultaneous set and clear on output 4	0
			00b no change	
			01b set output (or clear based on the SETCLR4 field in the OUTPUTDIRCTRL register)	
			10b set output (or clear based on the SETCLR4 field in the OUTPUTDIRCTRL register)	
			11b toggle output	
7:6	O3RES	R/W	effect of simultaneous set and clear on output 3	0
			00b no change	
			01b set output (or clear based on the SETCLR3 field in the OUTPUTDIRCTRL register)	
			10b set output (or clear based on the SETCLR3 field in the OUTPUTDIRCTRL register)	
			11b toggle output	
5:4	O2RES	R/W	effect of simultaneous set and clear on output 2	0
			00b no change	
			01b set output (or clear based on the SETCLR2 field in the OUTPUTDIRCTRL register)	
			10b set output (or clear based on the SETCLR2 field in the OUTPUTDIRCTRL register)	
			11b toggle output	
3:2	O1RES	R/W	effect of simultaneous set and clear on output 1	0
			00b no change	
			01b set output (or clear based on the SETCLR1 field in the OUTPUTDIRCTRL register)	
			10b set output (or clear based on the SETCLR1 field in the OUTPUTDIRCTRL register)	
			11b toggle output	

Table 806. SCT conflict resolution register - Offset: 58h...continued

Bit	Name	Access	Description and value	Reset value
1:0	OORES	R/W	effect of simultaneous set and clear on output 0	0
			00b no change	
			01b set output (or clear based on the SETCLR0 field in the OUTPUTDIRCTRL register)	
			10b set output (or clear based on the SETCLR0 field in the OUTPUTDIRCTRL register)	
			11b toggle output	

45.5.2.2.14 SCT DMA request 0 register (DMAREQ0)

Table 807. SCT DMA request 0 register - Offset: 5Ch

Bit	Name	Access	Description and value	Reset value
31	DRQ0	R	This read-only bit indicates the state of DMA request 0. Note: if the related DMA channel is enabled and properly set up, it is unlikely that software sees this flag. it is cleared quickly by the DMA service. The flag remaining set can point to an issue with DMA setup.	0
30	DRL0	R/W	A 1 in this bit triggers DMA request 0 when it loads the MATCH_L/ Unified registers from the RELOAD_L/ Unified registers.	0
29:16	reserved	-	reserved	-
15:0	DEV_0	R/W	If bit n is one, event n triggers DMA request 0 (event 0 = bit 0, event 1 = bit 1, and so on). The number of bits = the number of events in this SCT.	0

45.5.2.2.15 SCT DMA request 1 register (DMAREQ1)

Table 808. SCT DMA request 1 register - Offset: 60h

Bit	Name	Access	Description and value	Reset value
31	DRQ1	R	This read-only bit indicates the state of DMA request 1. Note: if the related DMA channel is enabled and properly set up, it is unlikely that software sees this flag. it is cleared quickly by the DMA service. The flag remaining set can point to an issue with DMA setup.	0
30	DRL1	R/W	A 1 in this bit triggers DMA request 0 when it loads the MATCH_L/ Unified registers from the RELOAD_L/ Unified registers.	0
29:16	reserved	-	reserved	-
15:0	DEV_1	R/W	If bit n is one, event n triggers DMA request 0 (event 0 = bit 0, event 1 = bit 1, and so on). The number of bits = the number of events in this SCT.	0

45.5.2.2.16 SCT event interrupt enable register (EVEN)

Table 809. SCT event interrupt enable register - Offset: F0h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved	-
15:0	IEN	R/W	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, and so on). The number of bits = the number of events in this SCT.	0

45.5.2.2.17 SCT event flag register (EVFLAG)

Table 810. SCT event flag register- Offset: F4h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved	-
15:0	FLAG	R/W	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, and so on). The number of bits = the number of events in this SCT.	0

45.5.2.2.18 SCT conflict interrupt enable register (CONEN)

Table 811. SCT conflict interrupt enable register - Offset: F8h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved	-
15:0	NCEN	R/W	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, and so on). The number of bits = the number of outputs in this SCT.	0

45.5.2.2.19 SCT conflict flag register (CONFLAG)

Table 812. SCT conflict flag register - Offset: FCh

Bit	Name	Access	Description and value	Reset value
31	BUSERRH	R/W	bus error high 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
30	BUSERRL	R/W	bus error low 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
29:16	reserved	-	reserved	-
15:0	NCFLAG	R/W	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, and so on). The number of bits = the number of outputs in this SCT.	0

45.5.2.2.20 SCT capture register of capture channel n (CAPn)

For n = 0 to 15

Table 813. SCT capture register of capture channel n - Offset: 100h + (n × 4h)

Bit	Name	Access	Description and value	Reset value
31:16	CAPn_H	R/W	capture high 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
15:0	CAPn_L	R/W	capture low 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

45.5.2.2.21 SCT match value register of match channel n (MATCHn)

For n = 0 to 15

Table 814. SCT match value register of match channel n - Offset: 100h + (n × 4h)

Bit	Name	Access	Description and value	Reset value
31:16	MATCHn_H	R/W	match high 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
15:0	MATCHn_L	R/W	match low 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

45.5.2.2.22 SCT capture control register n (CAPCTRLn)

For n = 0 to 15

Table 815. SCT capture control register n - Offset: 200h + (n × 4h)

Bit	Name	Access	Description and value	Reset value
31:16	CAPCONn_H	R/W	capture control high 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, and so on). The number of bits = the number of match/captures in this SCT.	0
15:0	CAPCONn_L	R/W	capture control low 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, and so on.). The number of bits = the number of match/captures in this SCT.	0

45.5.2.2.23 SCT match reload value register n (MATCHRELn)

For n = 0 to 15

Table 816. SCT match reload value register n - Offset: 200h + (n × 4h)

Bit	Name	Access	Description and value	Reset value
31:16	RELOADn_H	R/W	match reload high 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
15:0	RELOADn_L	R/W	match reload low 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

45.5.2.2.24 SCT event state register n (EVn_STATE)

For n = 0 to 15

Table 817. SCT event state register n - Offset: 300h + (n × 4h)

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved	-
15:0	STATMSKn	R/W	If bit m is one, event n happens in state m of the counter selected by the HEVENT bit (n = event number, m = state number; state 0 = bit 0, state 1 = bit 1, and so on). The number of bits = the number of states in this SCT.	0

45.5.2.2.25 SCT event control register n (EVn_CTRL)

For n = 0 to 15

Table 818. SCT event control register n - Offset: 304h + (n × 4h)

Bit	Name	Access	Description and value	Reset value						
31:23	reserved	-	reserved	-						
22:21	DIRECTION	R/W	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. <table><tr><td>00b</td><td>direction-independent This event is triggered regardless of the count direction.</td></tr><tr><td>01b</td><td>counting up This event is triggered only during up-counting when BIDIR = 1.</td></tr><tr><td>10b</td><td>counting down This event is triggered only during down-counting when BIDIR = 1.</td></tr></table>	00b	direction-independent This event is triggered regardless of the count direction.	01b	counting up This event is triggered only during up-counting when BIDIR = 1.	10b	counting down This event is triggered only during down-counting when BIDIR = 1.	0
00b	direction-independent This event is triggered regardless of the count direction.									
01b	counting up This event is triggered only during up-counting when BIDIR = 1.									
10b	counting down This event is triggered only during down-counting when BIDIR = 1.									

Table 818. SCT event control register n - Offset: 304h + (n × 4h)...continued

Bit	Name	Access	Description and value	Reset value
20	MATCHMEM	R/W	If this bit is one and the COMBMODE field specifies a match component to the triggering of this event, a match is considered to be active whenever the counter value ≥ the value specified in the match register when counting up; ≤ the match value when counting down. If this bit is zero, a match is only active during the cycle when the counter is equal to the match value.	0
19:15	STATEV	R/W	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
14	STATELD	R/W	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
			0b STATEV value is added into STATE (the carry-out is ignored)	
			1b STATEV value is loaded into STATE	
13:12	COMBMODE	R/W	selects how the specified match and I/O condition are used and combined	0
			00b OR The event occurs when either the specified match or I/O condition occurs.	
			01b MATCH Only uses the specified match.	
			10b IO Only uses the specified I/O condition.	
			11b AND The event occurs when the specified match and I/O condition occur simultaneously.	
11:10	IOCOND	R/W	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) To guarantee proper edge/state detection, an input must have a minimum pulse width of at least one SCT clock period .	0
			00b LOW	
			01b rise	
			10b fall	
			11b HIGH	
9:6	IOSEL	R/W	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
5	OUTSEL	R/W	input/output select	0
			0b selects the inputs selected by IOSEL	
			1b Selects the outputs selected by IOSEL	

Table 818. SCT event control register n - Offset: 304h + (n × 4h)...continued

Bit	Name	Access	Description and value	Reset value
4	HEVENT	R/W	select L/H counter Do not set this bit if UNIFY = 1.	0
			0b selects the L state and the L match register selected by MATCHSEL	
			1b selects the H state and the H match register selected by MATCHSEL	
3:0	MATCHSEL	R/W	selects the Match register associated with this event (if any) A match can only occur when the counter selected by the HEVENT bit is running.	0

45.5.2.2.26 SCT output n set register (OUTn_SET)

For n = 0 to 9

Table 819. SCT output n set register - Offset: 500h + (n × 4h)

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved	-
15:0	SET	R/W	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, and so on. The number of bits = the 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 OUTPUT register).	0

45.5.2.2.27 SCT output n clear register (OUTn_CLR)

For n = 0 to 9

Table 820. SCT output n clear register - Offset: 504h + (n × 4h)

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved	-
15:0	CLR	R/W	A 1 in bit m selects event m to clear output n (or clear it if SETCLRn = 0x1 or 0x2); output 0 = bit 0, output 1 = bit 1, and so on. The number of bits = the 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 OUTPUT register).	0

45.5.3 Flexcomm interface

45.5.3.1 Flexcomm memory map

The FLEXCOMM0 registers are located at base address: 4008_6000h.

The FLEXCOMM1 registers are located at base address: 4008_7000h.

The FLEXCOMM2 registers are located at base address: 4008_8000h

Table 821. Flexcomm serial communication registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
PSELID	R/W	FF8h	peripheral select and flexcomm ID register	32		Section 45.5.3.2.1
PID	R	FFCh	peripheral identification register	32	0000_0000h	Section 45.5.3.2.2

45.5.3.2 Register descriptions

45.5.3.2.1 Peripheral select and flexcomm ID register (PSELID)

Table 822. Peripheral select and flexcomm ID register - Offset: FF8h

Bit	Name	Access	Description and value	Reset value
31:12	ID	R	Flexcomm ID	0
11:7	reserved	-	reserved Read value is undefined, only zero must be written.	-
6	I2CPRESENT	R	I2C present indicator	1
			0b This Flexcomm does not include the I2C function	
			1b This Flexcomm includes the I2C function	
5	SPIPRESENT	R	SPI present indicator	1
			0b This Flexcomm does not include the SPI function	
			1b This Flexcomm includes the SPI function	
4	USARTPRESENT	R	USART present indicator	1
			0b This Flexcomm does not include the USART function	
			1b This Flexcomm includes the USART function	
3	LOCK	R/W	lock the peripheral select This field is writable by software.	0
			0b Peripheral select can be changed by software.	
			1b Peripheral select is locked and cannot be changed until this Flexcomm or the entire device is reset	

Table 822. Peripheral select and flexcomm ID register - Offset: FF8h...continued

Bit	Name	Access	Description and value	Reset value
2:0	PERSEL	R/W	peripheral select This field is writable by software.	0
			000b no peripheral selected	
			001b USART function selected	
			010b SPI function selected	
			011b I2C function selected	
			1xxb reserved	

45.5.3.2.2 Peripheral identification register (PID)

Table 823. Peripheral identification register - Offset: FFCh

Bit	Name	Access	Description and value	Reset value
31:16	ID	R	module identifier for the selected function	0
15:12	MAJOR_REV	R	major revision of module implementation	0
11:8	MINOR_REV	R	Minor revision of module implementation	0
7:0	reserved	-	reserved	-

45.5.4 USART

45.5.4.1 USART memory map

The USART0 registers are located at base address: 4008_6000h.

The USART1 registers are located at base address: 4008_7000h

The USART2 registers are located at base address: 4008_8000h

Table 824. Flexcomm serial communication registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CFG	R/W	0h	USART configuration register	32		Section 45.5.4.2.1
CTL	R/W	4h	USART control register	32		Section 45.5.4.2.2
STAT	W	8h	USART status register	32		Section 45.5.4.2.3
INTENSET	R/W	Ch	interrupt enable read and set register for USART (not FIFO) status	32		Section 45.5.4.2.4
INTENCLR	W	10h	interrupt enable clear register	32		Section 45.5.4.2.5
BRG	R/W	20h	baud rate generator register	32		Section 45.5.4.2.6
INTSTAT	R	24h	interrupt status register	32		Section 45.5.4.2.7
OSR	R/W	28h	oversample selection register for asynchronous communication	32		Section 45.5.4.2.8
ADDR	R/W	2Ch	address register for automatic address matching	32		Section 45.5.4.2.9
FIFOCFG	R/W	E00h	FIFO configuration and enable register	32		Section 45.5.4.2.10
FIFOSTAT	R/W	E04h	FIFO status register	32		Section 45.5.4.2.11
FIFOTRIG	R/W	E08h	FIFO trigger settings for interrupt and DMA request	32		Section 45.5.4.2.12
FIFOINTENSET	R/W	E10h	FIFO interrupt enable set (enable) and read register	32		Section 45.5.4.2.13
FIFOINTENCLR	R/W	E14h	FIFO interrupt enable clear (disable) and read register	32		Section 45.5.4.2.14
FIFOINTSTAT	R/W	E18h	FIFO interrupt status register	32		Section 45.5.4.2.15
FIFOWR	W	E20h	FIFO write data	32		Section 45.5.4.2.16
FIFORD	R	E30h	FIFO read data	32		Section 45.5.4.2.17
FIFORDNOPOP	R	E40h	FIFO data read with no FIFO pop	32		Section 45.5.4.2.18

Table 824. Flexcomm serial communication registers...continued

Name	Access	Offset	Description	Width (bits)	Reset value	Section
FIFOSIZE	W	E48h	FIFO size register	32	0000_0008h	Section 45.5.4.2.19

45.5.4.2 Register descriptions

45.5.4.2.1 USART configuration register (CFG)

Basic USART configuration settings that typically are not changed during operation.

Table 825. USART configuration register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:24	reserved	-	reserved Read value is undefined, only zero must be written.	-
23	TXPOL	R/W	Transmit data polarity 0b 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. 1b 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.	0
22	RXPOL	R/W	Receive data polarity 0b 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. 1b 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.	0
21	OEPOL	R/W	output enable polarity 0b low If selected by OESEL, the output enable is active low. 1b high If selected by OESEL, the output enable is active high.	0
20	OESEL	R/W	output enable select 0b standard The RTS signal is used as the standard flow control function. 1b RS-485 The RTS signal is configured to provide an output enable signal to control an RS-485 transceiver.	0

Table 825. USART configuration register - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value
19	AUTOADDR	R/W	automatic address matching enable	0
			0b disabled When addressing is enabled by ADDRDET, address matching is done by software. It provides the possibility of versatile addressing (for example, respond to more than one address).	
			1b 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.	
18	OETA	R/W	output enable turnaround time enable for RS-485 operation	0
			0b disabled If selected by OESEL, the output enable signal is deasserted at the end of the last stop bit of a transmission.	
			1b 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 also remains asserted if another transmit begins before it is deasserted.	
17:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15	LOOP	R/W	selects data loopback mode	0
			0b normal operation	
			1b loopback mode 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 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.	
14	SYNCMST	R/W	synchronous mode master select	0
			0b slave When synchronous mode is enabled, the USART is a slave.	
			1b master When synchronous mode is enabled, the USART is a master.	
13	reserved	-	reserved Read value is undefined, only zero must be written.	-

Table 825. USART configuration register - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value
12	CLKPOL	R/W	selects the clock polarity and sampling edge of received data in synchronous mode	0
			0b falling edge Un_RXD is sampled on the falling edge of SCLK.	
			1b rising edge Un_RXD is sampled on the rising edge of SCLK.	
11	SYNCEN	R/W	selects synchronous or asynchronous operation	0
			0b asynchronous mode	
			1b synchronous mode	
10	reserved	-	reserved Read value is undefined, only zero must be written.	-
9	CTSEN	R/W	CTS enable Determines if CTS is used for flow control. CTS can be used from the input pin or from the RTS of the USART if loopback mode is enabled.	0
			0b no flow control The transmitter does not receive any automatic flow control signal.	
			1b flow controle enabled The transmitter uses the CTS input (or RTS output in loopback mode) for flow control purposes.	
8	LINMODE	R/W	LIN break mode enable	0
			0b disabled Break detect and generate is configured for normal operation.	
			1b enabled Break detect and generate is configured for LIN bus operation.	
7	MODE32K	R/W	selects standard or 32 kHz clocking mode	0
			0b disabled USART uses standard clocking.	
			1b enabled USART uses the 32 kHz clock from the RTC oscillator as the clock source to the BRG. And it uses a special bit clocking scheme.	
6	STOPLEN	R/W	number of stop bits appended to transmitted data Only a single stop bit is required for received data.	0
			0b 1 stop bits	
			1b 2 stop bits Only use this setting for asynchronous communication.	

Table 825. USART configuration register - Offset: 0h...continued

Bit	Name	Access	Description and value		Reset value
5:4	PARITYSEL	R/W	selects what type of parity is used by the USART		0
			00b	no parity	
			01b	reserved	
			10b	even parity Adds a bit to each character such that the number of 1s in a transmitted character is even and they are in a received character is expected to be even.	
			11b	odd parity Adds a bit to each character such that the number of 1s in a transmitted character is odd, and they are in a received character is expected to be odd.	
3:2	DATALEN	R/W	selects the data size for the USART		0
			00b	7 bit data length	
			01b	8 bit data length	
			10b	9 bit data length The 9th bit is commonly used for addressing in multidrop mode. See the ADDRDET bit in Section 45.5.4.2.2 .	
			11b	reserved	
1	reserved	-	reserved Read value is undefined, only zero must be written.		-
0	ENABLE	R/W	USART enable		0
			0b	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 reenabled, the USART is ready to transmit immediately because the transmitter has been reset and is available.	
			1b	enabled The USART is enabled for operation.	

45.5.4.2.2 USART control register (CTL)

USART control settings that are more likely to change during operation.

Table 826. USART control register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:17	reserved	-	reserved Read value is undefined, only zero must be written.	-
16	AUTOBAUD	R/W	Autobaud enable	0
			0b disabled USART is in normal operating mode.	
			1b enabled USART is in autobaud mode. This bit must only be set when the USART receiver is idle. The first start bit of RX is measured and used to update the BRG register to match the received data rate. When this process is complete or if there is an AERR, AUTOBAUD is cleared.	
15:10	reserved	-	reserved Read value is undefined, only zero must be written.	-
9	CLRCONRX	R/W	clear continuous clock	0
			0b no effect No effect on the CC bit.	
			1b autoclear When a complete character has been received, the CC bit is automatically cleared. This bit is cleared at the same time.	
8	CC	R/W	continuous clock generation By default, SCLK is only output while data is transmitted in synchronous mode.	0
			0b clock on character In synchronous mode, SCLK cycles only when characters are being sent on Un_TXD or to complete a character that is received.	
			1b continuous clock SCLK runs continuously in synchronous mode, allowing characters to be received on Un_RxD independently from transmission on Un_TXD).	
7	reserved	-	reserved Read value is undefined, only zero must be written.	-
6	TXDIS	R/W	transmit disable	0
			0b not disabled USART transmitter is not disabled.	
			1b disabled USART transmitter is disabled after any character currently transmitted is complete. This feature can be used to facilitate software flow control.	

Table 826. USART control register - Offset: 4h...continued

Bit	Name	Access	Description and value	Reset value
5:3	reserved	-	reserved Read value is undefined, only zero must be written.	-
2	ADDRDET	R/W	enable address detect mode 0b disabled The USART presents all incoming data. 1b 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 must be handled. If it is, the ADDRDET bit is cleared by software and further incoming data is handled normally.	0
1	TXBRKEN	R/W	break enable 0b normal operation 1b 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.	0
0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.5.4.2.3 USART status register (STAT)

The complete status value can be read here. Writing ones clears some bits in the register.

Table 827. USART status register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:17	reserved	-	reserved Read value is undefined, only zero must be written.	-
16	ABERR	W	autobaud error An autobaud error can occur if the BRG counts to its limit before the end of the start bit that is measured; essentially an autobaud timeout.	0
15	RXNOISEINT	W	received noise interrupt flag Three samples of received data are taken to determine the value of each received data bit, except in synchronous mode. It acts as a noise filter if one sample disagrees. This flag is set when a received data bit contains one disagreeing sample. It can indicate line noise, a baud rate or character format mismatch, or loss of synchronization during data reception.	-

Table 827. USART status register - Offset: 8h...continued

Bit	Name	Access	Description and value	Reset value
14	PARITYERRINT	W	parity error interrupt flag This flag is set when a parity error is detected in a received character.	0
13	FRAMERRINT	W	framing error interrupt flag This flag is set when a character is received with a missing stop bit at the expected location. This can be an indication of a baud rate or configuration mismatch with the transmitting source.	0
12	START	W	This bit is set when a start is detected on the receiver input. Its purpose is primarily to allow wake-up from deep-sleep or power-down mode immediately when a start is detected. Cleared by software.	-
11	DELTARXBRK	W	This bit is set when a change in the state of receiver break detection occurs. Cleared by software.	0
10	RXBRK	R	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: FRAMERRINT is also set when this condition occurs because one or more the stop bit for the character are missing. RXBRK is cleared when the Un_RXD pin goes high.	0
9:7	reserved	-	reserved Read value is undefined, only zero must be written.	-
6	TXDISSTAT	R	transmitter disabled status flag When 1, this bit indicates that the USART transmitter is fully idle after it is disabled via the TXDIS bit in the CFG register (TXDIS = 1).	0
5	DELTACTS	W	This bit is set when a change in the state is detected for the CTS flag above. Cleared by software.	0
4	CTS	R	This bit reflects the current state of the CTS signal, regardless of the setting of the CTSEN bit in the CFG register. It is the value of the CTS input pin unless loopback mode is enabled.	0
3	TXIDLE	R	transmitter idle When 0, indicates that the transmitter is currently in the process of sending data. When 1, indicates that the transmitter is not currently in the process of sending data.	1
2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	RXIDLE	R	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
0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.5.4.2.4 Interrupt enable read and set register for USART (not FIFO) status (INTENSET)

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.

Table 828. Interrupt enable read and set register for USART (not FIFO) status - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:17	reserved	-	reserved Read value is undefined, only zero must be written.	-
16	ABERREN	R/W	When 1, enables an interrupt when an autobaud error occurs.	0
15	RXNOISEEN	R/W	When 1, enables an interrupt when noise is detected. See description of the RXNOISEINT bit in Section 45.5.4.2.3 .	0
14	PARITYERREN	R/W	When 1, enables an interrupt when a parity error has been detected.	0
13	FRAMERREN	R/W	When 1, enables an interrupt when a framing error has been detected.	0
12	STARTEN	R/W	When 1, enables an interrupt when a received start bit has been detected.	-
11	DELTARXBRKEN	R/W	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
10:7	reserved	-	reserved Read value is undefined, only zero must be written.	-
6	TXDISEN	R/W	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 in Section 45.5.4.2.7 for details.	0
5	DELTACTSEN	R/W	When 1, enables an interrupt when there is a change in the state of the CTS input.	0
4	reserved	-	reserved Read value is undefined, only zero must be written.	-
3	TXIDLEEN	R/W	When 1, enables an interrupt when the transmitter becomes idle (TXIDLE = 1).	1
2:0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.5.4.2.5 Interrupt enable clear register (INTENCLR)

Allows clearing any combination of bits in the INTENSET register. Writing a 1 to any implemented bit position causes the corresponding bit to be cleared.

Table 829. Interrupt enable clear register - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:17	reserved	-	reserved Read value is undefined, only zero must be written.	-
16	ABERRCLR	W	Writing 1 clears the corresponding bit in the INTENSET register.	0
15	RXNOISECLR	W	Writing 1 clears the corresponding bit in the INTENSET register.	0

Table 829. Interrupt enable clear register - Offset: 10h...continued

Bit	Name	Access	Description and value	Reset value
14	PARITYERRCLR	W	Writing 1 clears the corresponding bit in the INTENSET register.	0
13	FRAMERRCLR	W	Writing 1 clears the corresponding bit in the INTENSET register.	0
12	STARTCLR	W	Writing 1 clears the corresponding bit in the INTENSET register.	-
11	DELTARXBRKCLR	W	Writing 1 clears the corresponding bit in the INTENSET register.	0
10:7	reserved	-	reserved Read value is undefined, only zero must be written.	-
6	TXDISCLR	W	Writing 1 clears the corresponding bit in the INTENSET register.	0
5	DELTACTSCLR	W	Writing 1 clears the corresponding bit in the INTENSET register.	0
4	reserved	-	reserved Read value is undefined, only zero must be written.	-
3	TXIDLECLRE	W	Writing 1 clears the corresponding bit in the INTENSET register.	1
2:0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.5.4.2.6 Baud rate generator register (BRG)

16-bit integer baud rate divisor value.

Table 830. Baud rate generator register - Offset: 20h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15:0	BRGVAL	R/W	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

45.5.4.2.7 Interrupt status register (INTSTAT)

Reflects interrupts that are currently enabled.

Table 831. Interrupt status register - Offset: 24h

Bit	Name	Access	Description and value	Reset value
31:17	reserved	-	reserved Read value is undefined, only zero must be written.	-
16	ABERRINT	R	autobaud error Interrupt flag	0
15	RXNOISEINT	R	received noise interrupt flag	0

Table 831. Interrupt status register - Offset: 24h...continued

Bit	Name	Access	Description and value	Reset value
14	PARITYERRINT	R	parity error interrupt flag	0
13	FRAMERRINT	R	Framing Error interrupt flag	0
12	START	R	This bit is set when a start is detected on the receiver input.	0
11	DELTARXBRK	R	This bit is set when a change in the state of receiver break detection occurs.	0
10:7	reserved	-	reserved Read value is undefined, only zero must be written.	-
6	TXDISINT	R	transmitter disabled interrupt flag	0
5	DELTACTS	R	This bit is set when a change in the state of the CTS input is detected.	0
4	reserved	-	reserved Read value is undefined, only zero must be written.	0
3	TXIDLE	R	transmitter idle status	0
2:0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.5.4.2.8 Oversample selection register for asynchronous communication (OSR)

Table 832. Oversample selection register for asynchronous communication - Offset: 28h

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved Read value is undefined, only zero must be written.	-
3:0	OSRVAL	R/W	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.	0

45.5.4.2.9 Address register for automatic address matching (ADDR)

Table 833. Address register for automatic address matching - Offset: 2Ch

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	ADDRESS	R/W	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

45.5.4.2.10 FIFO configuration and enable register (FIFOCFG)

Table 834. FIFO configuration and enable register - Offset: E00h

Bit	Name	Access	Description and value	Reset value
31:19	reserved	-	reserved Read value is undefined, only zero must be written.	-
18	POPDBG	R/W	pop FIFO for debug reads	0
			0b debug reads of the FIFO do not pop the FIFO	
			1b a debug read causes the FIFO to pop	
17	EMPTYRX	R/W	empty command for the receive FIFO When a 1 is written to this bit, the RX FIFO is emptied.	0
16	EMPTYTX	R/W	empty command for the transmit FIFO When a 1 is written to this bit, the TX FIFO is emptied.	0
15	WAKERX	R/W	wake-up for receive FIFO level Allows the device to be woken from reduced power modes (up to power-down, 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 remains stopped until woken by another cause, such as DMA completion. See hardware wake-up control register.	0
			0b Only enabled interrupts wake up the device from reduced power modes.	
			1b A device wake-up for DMA occurs if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.	
14	WAKETX	R/W	wake-up for transmit FIFO level Allows the device to be woken from reduced power modes (up to power-down, 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 remains stopped until woken by another cause, such as DMA completion. See hardware wake-up control register.	0
			0b Only enabled interrupts wake up the device from reduced power modes.	
			1b A device wake-up for DMA occurs if the transmit FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.	
13	DMARX	R/W	DMA configuration for receive	0
			0b DMA is not used for the receive function.	
			1b Trigger DMA for the receive function if the FIFO is not empty. Generally, data interrupts is disabled if DMA is enabled.	

Table 834. FIFO configuration and enable register - Offset: E00h...continued

Bit	Name	Access	Description and value	Reset value
12	DMATX	R/W	DMA configuration for transmit	0
			0b DMA is not used for the transmit function.	
			1b Trigger DMA for the transmit function if the FIFO is not full. Generally, data interrupts is disabled if DMA is enabled.	
11:6	reserved	-	reserved Read value is undefined, only zero must be written.	-
5:4	SIZE	R	FIFO size configuration Read-only field. 0x0 = FIFO is configured as 16 entries of 8 bits. 0x1, 0x2, 0x3 = not applicable to USART.	0
3:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	ENABLERX	R/W	enable the receive FIFO	0
			0b receive FIFO is not enabled	
			1b receive FIFO is enabled	
0	ENABLETX	R/W	enable the transmit FIFO	0
			0b transmit FIFO is not enabled	
			1b transmit FIFO is enabled	

45.5.4.2.11 FIFO status register (FIFOSTAT)

Table 835. FIFO status register - Offset: E04h

Bit	Name	Access	Description and value	Reset value
31:21	reserved	-	reserved Read value is undefined, only zero must be written.	-
20:16	RXLVL	R	receive FIFO current level A 0 means the RX FIFO is currently empty and the RXFULL and RXNOTEMPTY flags are 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 are 1.	0
15:13	reserved	-	reserved Read value is undefined, only zero must be written.	-
12:8	TXLVL	R	Transmit FIFO current level A 0 means the TX FIFO is currently empty and the TXEMPTY and TXNOTFULL flags are 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 are 0.	0
7	RXFULL	R	receive FIFO full When 1, the receive FIFO is full. To prevent the peripheral from causing an overflow, data must be read out.	0

Table 835. FIFO status register - Offset: E04h...continued

Bit	Name	Access	Description and value	Reset value
6	RXNOTEMPTY	R	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
5	TXNOTFULL	R	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 causes it to overflow.	1
4	TXEMPTY	R	transmit FIFO empty When 1, the transmit FIFO is empty. The peripheral can still process the last piece of data.	1
3	PERINT	R	peripheral interrupt When 1, it indicates that the peripheral function has asserted an interrupt. The details can be found by reading the STAT register of the peripheral.	0
2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	RXERR	R	RX FIFO error Is 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
0	TXERR	R	TX FIFO error Is set if a transmit FIFO error occurs. This can an overflow caused by pushing data into a full FIFO. Or it can be an underflow if the FIFO is empty when data is required. Cleared by writing a 1 to this bit.	0

45.5.4.2.12 FIFO trigger settings for interrupt and DMA request (FIFOTRIG)

Table 836. FIFO trigger settings for interrupt and DMA request - Offset: E08h

Bit	Name	Access	Description and value	Reset value
31:20	reserved	-	reserved Read value is undefined, only zero must be written.	-
19:16	RXLVL	R/W	receive FIFO level trigger point The RX FIFO level is checked when a new piece of data is received. This field is only used 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. See hardware wake-up control register. 0 = trigger when the RX FIFO has received one entry (is no longer empty). 1 = trigger when the RX FIFO has received two entries. 15 = trigger when the RX FIFO has received 16 entries (has become full).	0
15:12	reserved	-	reserved Read value is undefined, only zero must be written.	-

Table 836. FIFO trigger settings for interrupt and DMA request - Offset: E08h...continued

Bit	Name	Access	Description and value	Reset value
11:8	TXLVL	R/W	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. See hardware wake-up control register. 0 = trigger when the TX FIFO becomes empty. 1 = trigger when the TX FIFO level decreases to one entry. 15 = trigger when the TX FIFO level decreases to 15 entries (is no longer full).	0
7:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	RXLVENA	R/W	receive FIFO level trigger enable This trigger becomes an interrupt if enabled in FIFOINTENSET. Or it becomes a DMA trigger if DMARX in FIFOCFG is set.	0
			0b Receive FIFO level does not generate a FIFO level trigger.	
			1b A trigger is generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.	
0	TXLVLENA	R/W	Transmit FIFO level trigger enable This trigger becomes an interrupt if enabled in FIFOINTENSET. Or it becomes a DMA trigger if DMATX in FIFOCFG is set.	0
			0b Transmit FIFO level does not generate a FIFO level trigger.	
			1b A trigger is generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.	

45.5.4.2.13 FIFO interrupt enable set (enable) and read register (FIFOINTENSET)

Table 837. FIFO interrupt enable set (enable) and read register - Offset: E10h

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved Read value is undefined, only zero must be written.	-
3	RXLVL	R/W	Determines if an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0
			0b No interrupt is generated based on the RX FIFO level.	
			1b If RXLVLENA in the FIFOTRIG register = 1, an interrupt is generated when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.	
2	TXLVL	R/W	Determines if an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0
			0b No interrupt is generated based on the TX FIFO level.	
			1b If TXLVLENA in the FIFOTRIG register = 1, an interrupt is generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.	
1	RXERR	R/W	Determines if an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register.	0
			0b No interrupt is generated for a receive error.	
			1b An interrupt is generated when a receive error occurs.	
0	TXERR	R/W	Determines if an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register.	0
			0b No interrupt is generated for a transmit error.	
			1b An interrupt is generated when a transmit error occurs.	

45.5.4.2.14 FIFO interrupt enable clear (disable) and read register (FIFOINTENCLR)

Table 838. FIFO interrupt enable clear (disable) and read register - Offset: E14h

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved Read value is undefined, only zero must be written.	-
3	RXLVL	R/W	Writing one clears the corresponding bits in the FIFOINTENSET register.	0
2	TXLVL	R/W	Writing one clears the corresponding bits in the FIFOINTENSET register.	0
1	RXERR	R/W	Writing one clears the corresponding bits in the FIFOINTENSET register.	0
0	TXERR	R/W	Writing one clears the corresponding bits in the FIFOINTENSET register.	0

45.5.4.2.15 FIFO interrupt status register (FIFOINTSTAT)

Table 839. FIFO interrupt status register - Offset: E18h

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved Read value is undefined, only zero must be written.	-
4	PERINT	R	peripheral interrupt	0
3	RXLVL	R	receive FIFO level interrupt	0
2	TXLVL	R	transmit FIFO level interrupt	0
1	RXERR	R	RX FIFO error	0
0	TXERR	R	TX FIFO error	0

45.5.4.2.16 FIFO write data (FIFOWR)

Table 840. FIFO write data - Offset: E20h

Bit	Name	Access	Description and value	Reset value
31:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8:0	TXDATA	W	transmit data to the FIFO	0

45.5.4.2.17 FIFO read data (FIFORD)

Table 841. FIFO read data - Offset: E30h

Bit	Name	Access	Description and value	Reset value
31:15	reserved	-	reserved Read value is undefined, only zero must be written.	-
15	RXNOISE	R	received noise flag See description of the RxNoiseInt bit in Section 45.5.4.2.3 .	0
14	PARITYERR	R	parity error status flag This bit reflects the status for the data it is read along with from the FIFO. When a parity error is detected in a received character, this bit is set.	0
13	FRAMERR	R	framing error status flag This bit reflects the status for the data it is read along with from the FIFO. It also indicates that the character was received with a missing stop bit at the expected location. This can be an indication of a baud rate or configuration mismatch with the transmitting source.	0
12:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8:0	RXDATA	R	received data from the FIFO The number of bits used depends on the DATALEN and PARITYSEL settings.	0

45.5.4.2.18 FIFO data read with no FIFO pop (FIFORDNOPOP)

Table 842. FIFO data read with no FIFO pop - Offset: E40h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15	RXNOISE	R	received noise flag See description of the RxNoiseInt bit in Section 45.5.4.2.3 .	0
14	PARITYERR	R	parity error status flag This bit reflects the status for the data it is read along with from the FIFO. When a parity error is detected in a received character, this bit is set.	0
13	FRAMERR	R	framing error status flag This bit reflects the status for the data it is read along with from the FIFO. It also indicates that the character was received with a missing stop bit at the expected location. This can be an indication of a baud rate or configuration mismatch with the transmitting source.	0
12:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8:0	RXDATA	R	received data from the FIFO The number of bits used depends on the DATALEN and PARITYSEL settings.	0

45.5.4.2.19 FIFO size register (FIFOSIZE)

The FIFOSIZE register provides the size FIFO for the selected Flexcomm function on this device.

Table 843. FIFO size register - Offset: E48h

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved Read value is undefined, only zero must be written.	-
4:0	FIFOSIZE	W	provides the size of the FIFO for software The size of the UART FIFO is 8 entries.	0

45.5.5 Serial peripheral interfaces (SPI)

45.5.5.1 SPI memory map

The SPI0 registers are located at base address: 4008_6000h.

The SPI1 registers are located at base address: 4008_7000h.

The SPI2 registers are located at base address: 4008_8000h.

Table 844. Serial peripheral interface registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CFG	R/W	400h	SPI configuration register	32		Section 45.5.5.2.1
DLY	R/W	404h	SPI delay register	32		Section 45.5.5.2.2
STAT	R/W	408h	SPI status register	32		Section 45.5.5.2.3
INTENSET	R/W	40Ch	SPI interrupt enable read and set	32		Section 45.5.5.2.4
INTENCLR	W	410h	SPI interrupt enable clear	32		Section 45.5.5.2.5
DIV	R/W	424h	SPI clock divider	32		Section 45.5.5.2.6
INTSTAT	R	428h	SPI interrupt status	32		Section 45.5.5.2.7
FIFOCFG	R/W	E00h	FIFO configuration and enable register	32		Section 45.5.5.2.8
FIFOSTAT	R/W	E04h	FIFO status register	32		Section 45.5.5.2.9
FIFOTRIG	R/W	E08h	FIFO trigger settings for interrupt and DMA request	32		Section 45.5.5.2.10
FIFOINTENSET	R/W	E10h	FIFO interrupt enable set (enable) and read register	32		Section 45.5.5.2.11
FIFOINTENCLR	R/W	E14h	FIFO interrupt enable clear (disable) and read register	32		Section 45.5.5.2.12
FIFOINTSTAT	R/W	E18h	FIFO interrupt status register	32		Section 45.5.5.2.13
FIFOWR	W	E20h	FIFO write data	32		Section 45.5.5.2.14
FIFORD	R	E30h	FIFO read data	32		Section 45.5.5.2.15
FIFORDNOPOP	R	E40h	FIFO data read with no FIFO pop	32		Section 45.5.5.2.16
FIFOSIZE	W	E48h	FIFO size register	32	0000_0008h	Section 45.5.5.2.17

45.5.5.2 Register descriptions

45.5.5.2.1 SPI configuration register (CFG)

Table 845. SPI configuration register - Offset: 400h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved Read value is undefined, only zero must be written.	-
9	SPOL1	R/W	SSEL1 polarity select	0
			0b low The SSEL1 pin is active low.	
			1b high The SSEL1 pin is active high.	
8	SPOL1	R/W	SSEL0 polarity select	0
			0b low The SSEL0 pin is active low.	
			1b high The SSEL0 pin is active high.	
7	LOOP	R/W	output enable polarity	0
			0b low If selected by OESEL, the output enable is active low.	
			1b high If selected by OESEL, the output enable is active high.	
6	reserved	-	reserved Read value is undefined, only zero must be written.	-
5	CPOL	R/W	clock polarity select	0
			0b low The rest state of the clock (between transfers) is low.	
			1b high The rest state of the clock (between transfers) is high.	
4	CPHA	R/W	clock phase select	0
			0b 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.	
			1b 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.	

Table 845. SPI configuration register - Offset: 400h...continued

Bit	Name	Access	Description and value	Reset value
3	LSBF	R/W	LSB first mode enable	0
			0b standard Data is transmitted and received in standard MSB first order.	
			1b reverse Data is transmitted and received in reverse order (LSB first).	
2	MASTER	R/W	master mode select	0
			0b slave mode The SPI operates in slave mode. SCK, MOSI, and the SSEL signals are inputs, MISO is an output.	
			1b master mode The SPI operates in master mode. SCK, MOSI, and the SSEL signals are outputs, MISO is an input.	
1	reserved	-	reserved Read value is undefined, only zero must be written.	-
0	ENABLE	R/W	SPI enable	0
			0b disabled The SPI is disabled and the internal state machine and counters are reset.	
			1b enabled The SPI is enabled for operation.	

45.5.5.2.2 SPI delay register (DLY)

Table 846. SPI delay register - Offset: 404h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15:12	TRANSFER_DELAY	R/W	controls the minimum amount of time that the SSEL is deasserted between transfers 0x0 = the minimum time that SSEL is deasserted is 1 SPI clock time (Zero added time). 0x1 = the minimum time that SSEL is deasserted is 2 SPI clock times. 0x2 = the minimum time that SSEL is deasserted is 3 SPI clock times. 0xF = the minimum time that SSEL is deasserted is 16 SPI clock times.	0
11:8	FRAME_DELAY	R/W	If the EOF flag is set, controls the minimum amount of time between the current frame and the next frame (or SSEL deassertion 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

Table 846. SPI delay register - Offset: 404h...continued

Bit	Name	Access	Description and value	Reset value
7:4	POST_DELAY	R/W	controls the amount of time between the end of a data transfer and SSEL deassertion 0x0 = no additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. 0xF = 15 SPI clock times are inserted.	0
3:0	PRE_DELAY	R/W	controls the amount of time between SSEL assertion and the beginning of a data transfer There is always 1 SPI clock time between SSEL assertion and the first clock edge. It is not considered part of the predelay. 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

45.5.5.2.3 SPI status (STAT)

Table 847. SPI status - Offset: 408h

Bit	Name	Access	Description and value	Reset value
31:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8	MSTIDLE	R	master idle status flag This bit is 1 when the SPI master function is fully idle. The transmit holding register is empty then and the transmitter is not in the process of sending data.	1
7	ENDTRANSFER	R/W	end transfer control bit When the transmitter finishes any activity already in progress, software can set this bit to force an end to the current transfer. As if the EOT flag has been set prior to the last transmission. This capability is included to support cases where it is not known when transmit data is written is the end of a transfer. When the transmitter becomes idle as the transfer comes to an end, the bit is cleared. Forcing an end of transfer in this manner causes any specified FRAME_DELAY and TRANSFER_DELAY to be inserted.	0
6	STALLED	R	stalled status flag Indicates if the SPI is currently in a stall condition.	0
5	SSD	W	slave select deassert When any asserted slave selects transition to deasserted, in both master and slave modes, this flag is set. It allows the determination of when the SPI transmit/receive functions become idle. This flag is cleared by software.	0
4	SSA	W	slave select assert When any slave select transitions from deasserted to asserted, in both master and slave modes, this flag is set. It allows the determination of when the SPI transmit/receive functions become busy. It also allows the waking up of the device from reduced power modes when a slave mode access begins. This flag is cleared by software.	0

Table 847. SPI status - Offset: 408h...continued

Bit	Name	Access	Description and value	Reset value
3:0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.5.5.2.4 SPI interrupt enable read and set (INTENSET)

Table 848. SPI interrupt enable read and set - Offset: 40Ch

Bit	Name	Access	Description and value	Reset value
31:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8	MSTIDLEEN	R/W	master idle interrupt enable 0b No interrupt is generated when the SPI master function is idle. 1b An interrupt is generated when the SPI master function is fully idle.	0
7:6	reserved	-	reserved Read value is undefined, only zero must be written.	-
5	SSDEN	R/W	slave select deassert interrupt enable Determines if an interrupt occurs when the slave select is deasserted. 0b disabled No interrupt is generated when all asserted slave selects transition to deasserted. 1b enabled An interrupt is generated when all asserted slave selects transition to deasserted.	0
4	SSAEN	R/W	slave select deassert interrupt enable Determines if an interrupt occurs when the slave select is deasserted. 0b disabled No interrupt is generated when any asserted slave selects transitions from deasserted to asserted. 1b enabled An interrupt is generated when any asserted slave selects transitions from deasserted to asserted.	0
3:0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.5.5.2.5 SPI interrupt enable clear (INTENCLR)

Table 849. SPI interrupt enable clear - Offset: 410h

Bit	Name	Access	Description and value	Reset value
31:9	reserved	-	reserved Read value is undefined, only zero must be written.	-

Table 849. SPI interrupt enable clear - Offset: 410h...continued

Bit	Name	Access	Description and value	Reset value
8	MSTIDLE	W	Writing 1 clears the corresponding bit in the INTENSET register.	0
7:6	reserved	-	reserved Read value is undefined, only zero must be written.	-
5	SSDEN	W	Writing 1 clears the corresponding bit in the INTENSET register.	0
4	SSAEN	W	Writing 1 clears the corresponding bit in the INTENSET register.	0
3:0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.5.5.2.6 SPI clock divider (DIV)

Table 850. SPI clock divider - Offset: 424h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15:0	DIVVAL	R/W	rate divider value Specifies how the Flexcomm 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

45.5.5.2.7 SPI interrupt status (INTSTAT)

Table 851. SPI interrupt status - Offset: 428h

Bit	Name	Access	Description and value	Reset value
31:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8	MSTIDLE	R	master idle status flag	0
7:6	reserved	-	reserved Read value is undefined, only zero must be written.	-
5	SSD	R	slave select deassert	0
4	SSA	R	slave select assert	0
3:0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.5.5.2.8 FIFO configuration and enable register (FIFOCFG)

Table 852. FIFO configuration and enable register - Offset: E00h

Bit	Name	Access	Description and value	Reset value
31:19	reserved	-	reserved Read value is undefined, only zero must be written.	-
18	POPDBG	R/W	pop FIFO for debug reads	0
			0b debug reads of the FIFO do not pop the FIFO	
			1b a debug read causes the FIFO to pop	
17	EMPTYRX	R/W	empty command for the receive FIFO When a 1 is written to this bit, the RX FIFO is emptied.	0
16	EMPTYTX	R/W	empty command for the transmit FIFO When a 1 is written to this bit, the TX FIFO is emptied.	0
15	WAKERX	R/W	wake-up for receive FIFO level Allows the device to be woken from reduced power modes (up to power-down, 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 remains stopped until woken by another cause, such as DMA completion. See hardware wake-up control register.	0
			0b Only enabled interrupts wake up the device from reduced power modes.	
			1b A device wake-up for DMA occurs if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.	
14	WAKETX	R/W	wake-up for transmit FIFO level Allows the device to be woken from reduced power modes (up to power-down, 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 remains stopped until woken by another cause, such as DMA completion. See hardware wake-up control register.	0
			0b Only enabled interrupts wake up the device from reduced power modes.	
			1b A device wake-up for DMA occurs if the transmit FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.	
13	DMARX	R/W	DMA configuration for receive	0
			0b DMA is not used for the receive function.	
			1b Trigger DMA for the receive function if the FIFO is not empty. Generally, data interrupts is disabled if DMA is enabled.	

Table 852. FIFO configuration and enable register - Offset: E00h...continued

Bit	Name	Access	Description and value	Reset value
12	DMATX	R/W	DMA configuration for transmit	0
			0b DMA is not used for the transmit function.	
			1b Trigger DMA for the transmit function if the FIFO is not full. Generally, data interrupts is disabled if DMA is enabled.	
11:6	reserved	-	reserved Read value is undefined, only zero must be written.	-
5:4	SIZE	R	FIFO size configuration Read-only field. 0x0 = FIFO is configured as 16 entries of 8 bits. 0x1, 0x2, 0x3 = not applicable to USART.	0
3:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	ENABLERX	R/W	enable the receive FIFO	0
			0b receive FIFO is not enabled	
			1b receive FIFO is enabled	
0	ENABLETX	R/W	enable the transmit FIFO	0
			0b transmit FIFO is not enabled	
			1b transmit FIFO is enabled	

45.5.5.2.9 FIFO status register (FIFOSTAT)

Table 853. FIFO status register - Offset: E04h

Bit	Name	Access	Description and value	Reset value
31:21	reserved	-	reserved Read value is undefined, only zero must be written.	-
20:16	RXLVL	R	receive FIFO current level A 0 means the RX FIFO is currently empty and the RXFULL and RXNOTEMPTY flags are 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 are 1.	0
15:13	reserved	-	reserved Read value is undefined, only zero must be written.	-
12:8	TXLVL	R	Transmit FIFO current level A 0 means the TX FIFO is currently empty and the TXEMPTY and TXNOTFULL flags are 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 are 0.	0
7	RXFULL	R	receive FIFO full When 1, the receive FIFO is full. To prevent the peripheral from causing an overflow, data must be read out.	0

Table 853. FIFO status register - Offset: E04h...continued

Bit	Name	Access	Description and value	Reset value
6	RXNOTEMPTY	R	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
5	TXNOTFULL	R	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 causes it to overflow.	1
4	TXEMPTY	R	transmit FIFO empty When 1, the transmit FIFO is empty. The peripheral can still process the last piece of data.	1
3	PERINT	R	peripheral interrupt When 1, it indicates that the peripheral function has asserted an interrupt. The details can be found by reading the STAT register of the peripheral.	0
2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	RXERR	R	RX FIFO error Is 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
0	TXERR	R	TX FIFO error Is set if a transmit FIFO error occurs. This can an overflow caused by pushing data into a full FIFO. Or it can be an underflow if the FIFO is empty when data is required. Cleared by writing a 1 to this bit.	0

45.5.5.2.10 FIFO trigger settings for interrupt and DMA request (FIFOTRIG)

Table 854. FIFO trigger settings for interrupt and DMA request - Offset: E08h

Bit	Name	Access	Description and value	Reset value
31:20	reserved	-	reserved Read value is undefined, only zero must be written.	-
19:16	RXLVL	R/W	receive FIFO level trigger point The RX FIFO level is checked when a new piece of data is received. This field is only used 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. See hardware wake-up control register. 0 = trigger when the RX FIFO has received one entry (is no longer empty). 1 = trigger when the RX FIFO has received two entries. 15 = trigger when the RX FIFO has received 16 entries (has become full).	0
15:12	reserved	-	reserved Read value is undefined, only zero must be written.	-

Table 854. FIFO trigger settings for interrupt and DMA request - Offset: E08h...continued

Bit	Name	Access	Description and value	Reset value
11:8	TXLVL	R/W	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. See hardware wake-up control register. 0 = trigger when the TX FIFO becomes empty. 1 = trigger when the TX FIFO level decreases to one entry. 15 = trigger when the TX FIFO level decreases to 15 entries (is no longer full).	0
7:2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	RXLVENA	R/W	receive FIFO level trigger enable This trigger becomes an interrupt if enabled in FIFOINTENSET. Or it becomes a DMA trigger if DMARX in FIFOCFG is set.	0
			0b Receive FIFO level does not generate a FIFO level trigger.	
			1b A trigger is generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.	
0	TXLVLENA	R/W	Transmit FIFO level trigger enable This trigger becomes an interrupt if enabled in FIFOINTENSET. Or it becomes a DMA trigger if DMATX in FIFOCFG is set.	0
			0b Transmit FIFO level does not generate a FIFO level trigger.	
			1b A trigger is generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.	

45.5.5.2.11 FIFO interrupt enable set (enable) and read register (FIFOINTENSET)

Table 855. FIFO interrupt enable set (enable) and read register - Offset: E10h

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved Read value is undefined, only zero must be written.	-
3	RXLVL	R/W	Determines if an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0
			0b No interrupt is generated based on the RX FIFO level.	
			1b If RXLVLENA in the FIFOTRIG register = 1, an interrupt is generated when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.	
2	TXLVL	R/W	Determines if an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0
			0b No interrupt is generated based on the TX FIFO level.	
			1b If TXLVLENA in the FIFOTRIG register = 1, an interrupt is generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.	
1	RXERR	R/W	Determines if an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register.	0
			0b No interrupt is generated for a receive error.	
			1b An interrupt is generated when a receive error occurs.	
0	TXERR	R/W	Determines if an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register.	0
			0b No interrupt is generated for a transmit error.	
			1b An interrupt is generated when a transmit error occurs.	

45.5.5.2.12 FIFO interrupt enable clear (disable) and read register (FIFOINTENCLR)

Table 856. FIFO interrupt enable clear (disable) and read register - Offset: E14h

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved Read value is undefined, only zero must be written.	-
3	RXLVL	R/W	Writing one clears the corresponding bits in the FIFOINTENSET register.	0
2	TXLVL	R/W	Writing one clears the corresponding bits in the FIFOINTENSET register.	0
1	RXERR	R/W	Writing one clears the corresponding bits in the FIFOINTENSET register.	0
0	TXERR	R/W	Writing one clears the corresponding bits in the FIFOINTENSET register.	0

45.5.5.2.13 FIFO interrupt status register (FIFOINTSTAT)

Table 857. FIFO interrupt status register - Offset: E18h

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved Read value is undefined, only zero must be written.	-
4	PERINT	R	peripheral interrupt	0
3	RXLVL	R	receive FIFO level interrupt	0
2	TXLVL	R	transmit FIFO level interrupt	0
1	RXERR	R	RX FIFO error	0
0	TXERR	R	TX FIFO error	0

45.5.5.2.14 FIFO write data (FIFOWR)

Table 858. FIFO write data - Offset: E20h

Bit	Name	Access	Description and value	Reset value
31:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8:0	TXDATA	W	transmit data to the FIFO	0

45.5.5.2.15 FIFO read data (FIFORD)

Table 859. FIFO read data - Offset: E30h

Bit	Name	Access	Description and value	Reset value
31:15	reserved	-	reserved Read value is undefined, only zero must be written.	-
15	RXNOISE	R	received noise flag See description of the RxNoiseInt bit in Section 45.5.4.2.3 .	0
14	PARITYERR	R	parity error status flag This bit reflects the status for the data it is read along with from the FIFO. When a parity error is detected in a received character, this bit is set.	0
13	FRAMERR	R	framing error status flag This bit reflects the status for the data it is read along with from the FIFO. It also indicates that the character was received with a missing stop bit at the expected location. This can be an indication of a baud rate or configuration mismatch with the transmitting source.	0
12:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8:0	RXDATA	R	received data from the FIFO The number of bits used depends on the DATALEN and PARITYSEL settings.	0

45.5.5.2.16 FIFO data read with no FIFO pop (FIFORDNOPOP)

Table 860. FIFO data read with no FIFO pop - Offset: E40h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15	RXNOISE	R	received noise flag See description of the RxNoiseInt bit in Section 45.5.4.2.3 .	0
14	PARITYERR	R	parity error status flag This bit reflects the status for the data it is read along with from the FIFO. When a parity error is detected in a received character, this bit is set.	0
13	FRAMERR	R	framing error status flag This bit reflects the status for the data it is read along with from the FIFO. It also indicates that the character was received with a missing stop bit at the expected location. This can be an indication of a baud rate or configuration mismatch with the transmitting source.	0
12:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8:0	RXDATA	R	received data from the FIFO The number of bits used depends on the DATALEN and PARITYSEL settings.	0

45.5.5.2.17 FIFO size register (FIFOSIZE)

The FIFOSIZE register provides the size FIFO for the selected Flexcomm function on this device.

Table 861. FIFO size register - Offset: E48h

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved Read value is undefined, only zero must be written.	-
4:0	FIFOSIZE	W	provides the size of the FIFO for software The size of the UART FIFO is 8 entries.	0

45.5.6 I2C-bus interfaces

45.5.6.1 I2C memory map

The I2C0 base address is 4008_6000h.

The I2C1 base address is 4008_7000h.

The I2C2 base address is 4008_8000h.

Table 862. I2C registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CFG	R/W	800h	configuration for shared functions	32		Section 45.5.6.2.1
STAT	R/W	804h	Status register for master, slave, and monitor functions	32		Section 45.5.6.2.2
INTENSET	R/W	808h	interrupt enable set and read register	32		Section 45.5.6.2.3
INTENCLR	W	80Ch	interrupt enable clear register	32		Section 45.5.6.2.4
TIMEOUT	R/W	810h	timeout value register	32		Section 45.5.6.2.5
CLKDIV	R/W	814h	clock predivider for the entire I2C interface	32		Section 45.5.6.2.6
INTSTAT	R	818h	interrupt status register for master, slave, and monitor functions	32		Section 45.5.6.2.7
MSTCTL	R/W	820h	master control register	32		Section 45.5.6.2.8
MSTTIME	R/W	824h	master timing configuration	32		Section 45.5.6.2.9
MSTDAT	R/W	828h	combined master receiver and transmitter data register	32		Section 45.5.6.2.10
SLVCTL	R/W	840h	slave control register	32		Section 45.5.6.2.11
SLVDAT	R/W	844h	combined slave receiver and transmitter data register	32		Section 45.5.6.2.12
SLVADR0	R/W	848h	slave address register	32		Section 45.5.6.2.13
SLVADR1 - SLVADR3	R/W	84Ch - 854h	slave address register	32		Section 45.5.6.2.14
SLVQUAL0	R/W	858h	slave qualification for address 0	32		Section 45.5.6.2.15
MONRXDAT	R	880h	monitor receiver data register	32		Section 45.5.6.2.16

45.5.6.2 Register descriptions

45.5.6.2.1 Configuration for shared functions (CFG)

Table 863. Configuration for shared functions - Offset: 800h

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved Read value is undefined, only zero must be written.	-
5	HSCAPABLE	R/W	high-speed mode capable enable Since high-speed mode alters the way I2C pins drive and filter, as well as the timing for certain I2C signalling, enabling high-speed mode applies to all functions, master, slave, and monitor. 0b fast mode plus The I2C interface supports standard mode, fast mode, and fast mode plus, to the extent that the pin electronics support these modes. Any changes that must be made to the pin controls, such as changing the drive strength or filtering, must be made by software via the IOCON register associated with each I2C pin. 1b high speed In addition to standard mode, fast mode, and fast mode plus, the I2C interface supports high-speed mode to the extent that the pin electronics support these modes (see section Section 33.5.2.2).	0
4	MONCLKSTR	R/W	monitor function clock stretching 0b disabled The monitor function does 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. 1b enabled The monitor function performs clock stretching to ensure that software or DMA can read all incoming data supplied by the monitor function.	0
3	TIMEOUTEN	R/W	I2C bus timeout enable 0b disabled The timeout function is disabled. 1b enabled The timeout function is enabled. Both types of timeout flags are generated and cause interrupts if they are enabled. Typically, only one timeout is used in a system.	0

Table 863. Configuration for shared functions - Offset: 800h...continued

Bit	Name	Access	Description and value	Reset value
2	MONEN	R/W	monitor enable When disabled, configuration settings for the Monitor function are not changed. However, the monitor function is internally reset.	0
			0b disabled The I2C monitor function is disabled.	
			1b enabled The I2C monitor function is enabled.	
1	SLVEN	R/W	slave enable When disabled, configurations settings for the slave function are not changed. However, the slave function is internally reset.	0
			0b disabled The I2C slave function is disabled.	
			1b enabled The I2C slave function is enabled.	
0	MSTEN	R/W	master enable When disabled, configurations settings for the master function are not changed. However, the master function is internally reset.	0
			0b disabled The I2C master function is disabled.	
			1b enabled The I2C master function is enabled.	

45.5.6.2.2 Status register for master, slave, and monitor functions (STAT)

Table 864. Status register for master, slave, and monitor functions - Offset: 804h

Bit	Name	Access	Description and value	Reset value
31:26	reserved	-	reserved Read value is undefined, only zero must be written.	-
25	SCLTIMEOUT	R/W	SCL timeout 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
			0b no timeout SCL low time has not caused a timeout.	
			1b timeout SCL low time has caused a timeout.	

Table 864. Status register for master, slave, and monitor functions - Offset: 804h...continued

Bit	Name	Access	Description and value		Reset value
24	EVENTTIMEOUT	R/W	Event timeout 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 timeout is created when the I2C-bus is idle.		0
			0b	no timeout I2C bus events have not caused a timeout.	
			1b	event timeout The time between I2C bus events has been longer than the time specified by the TIMEOUT register.	
23:20	reserved	-	reserved Read value is undefined, only zero must be written.		-
19	MONIDLE	R/W	monitor idle flag This flag is set when the monitor function sees the I2C bus change from active to inactive. It can be used by software to decide when to process data accumulated by the monitor function. This flag causes an interrupt when set if enabled via the INTENSET register. The flag can be cleared by writing a 1 to this bit.		0
			0b	not idle The I2C bus is not idle, or this flag has been cleared by software.	
			1b	idle The I2C bus has gone idle at least once since the last time this flag was cleared by software.	
18	MONACTIVE	R	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
			0b	inactive The monitor function considers the I2C bus to be inactive.	
			1b	active The monitor function considers the I2C bus to be active.	
17	MONOV	R/W	monitor overflow flag		0
			0b	no overrun Monitor data has not overrun.	
			1b	overrun A monitor data overrun has occurred. It can only happen when monitor clock stretching not enabled via the MON CLKSTR bit in the CFG register. Writing 1 to this bit clears the flag.	

Table 864. Status register for master, slave, and monitor functions - Offset: 804h...continued

Bit	Name	Access	Description and value	Reset value
16	MONRDY	R	monitor ready This flag is cleared when the MONRXDAT register is read.	0
			0b no data The monitor function does not currently have data available.	
			1b data waiting The monitor function has data waiting to be read.	
15	SLVDESEL	R/W	slave deselected flag This flag causes an interrupt when set if enabled via INTENSET. It can be cleared by writing a 1 to this bit.	0
			0b not deselected The slave function has not become deselected. It does not mean that it is currently selected. That information can be found in the SLVSEL flag.	
			1b deselected The slave function has become deselected. It is specifically caused by the SLVSEL flag changing from 1 to 0. See the description of SLVSEL for details on when that event occurs.	
14	SLVSEL	R	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
			0b not selected The slave function is not currently selected.	
			1b selected The slave function is currently selected.	

Table 864. Status register for master, slave, and monitor functions - Offset: 804h...continued

Bit	Name	Access	Description and value	Reset value
13:12	SLVIDX	R	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. It provides an identification of the address that was matched. More than one address can be matched, but only one match can be reported here.	0
			00b address 0 Slave address 0 was matched.	
			01b address 1 Slave address 1 was matched.	
			10b address 2 Slave address 2 was matched.	
			11b address 3 Slave address 3 was matched.	
11	SLVNOTSTR	R	slave not stretching Indicates when the slave function is stretching the I2C clock. It is required to gracefully invoke deep-sleep mode or power-down mode during slave operation. This read-only flag reflects the slave function status in real time.	1
			0b stretching The slave function is currently stretching the I2C bus clock. Deep-sleep mode or power-down mode cannot be entered at this time.	
			1b not stretching The slave function is not currently stretching the I2C bus clock. Deep-sleep mode or power-down mode can be entered at this time.	
10:9	SLVSTATE	R	slave state code Each value of this field indicates a specific required service for the Slave function. All other values are reserved (see Table 401 for state values and actions). Note: The occurrence of some states and how they are handled are affected by DMA mode and automatic operation modes.	0
			00b slave address Address plus R/W received. At least one of the four slave addresses has been matched by hardware.	
			01b slave receive Received data is available (slave receiver mode).	
			10b slave transmit Data can be transmitted (slave transmitter mode).	

Table 864. Status register for master, slave, and monitor functions - Offset: 804h...continued

Bit	Name	Access	Description and value		Reset value
8	SLVPENDING	R	slave pending Indicates that the slave function is waiting to continue communication on the I2C-bus and requires software service. This flag causes 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 if the I2C interface is in HSCAPABLE mode. When the I2C interface is configured to be HSCAPABLE, HS master codes are detected automatically. Due to the requirements of the HS I2C specification, slave addresses must also be detected automatically, since the address must be acknowledged before the clock can be stretched.		0
			0b	in progress The slave function does not currently require service.	
			1b	pending The Slave function requires service. Information on what is required can be found in the adjacent SLVSTATE field.	
7	reserved	-	reserved Read value is undefined, only zero must be written.		-
6	MSTSTSTPERR	R/W	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
			0b	no start/stop error has occurred	
			1b	The master function has experienced a start/stop error. A start/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 can be made or software can attempt to ensure that the bus has not stalled.	
5	reserved	-	reserved Read value is undefined, only zero must be written.		-
4	MSTARBLOSS	R/W	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
			0b	No arbitration loss has occurred.	
			1b	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 to attempt to gain control of the bus when it next becomes idle.	

Table 864. Status register for master, slave, and monitor functions - Offset: 804h...continued

Bit	Name	Access	Description and value	Reset value
3:1	MSTSTATE	R	master state code The master state code reflects the master state when the MSTPEN DING 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 400 for details of state values and appropriate responses.	0
			000b idle The master function is available to be used for a new transaction.	
			001b receive ready Received data available (master receiver mode). Address plus read was previously sent and acknowledged by slave.	
			010b transmit ready Data can be transmitted (master transmitter mode). Address plus write was previously sent and acknowledged by slave.	
			011b NACK address Slave NACKed address.	
			100b NACK data Slave NACKed transmitted data.	
0	MSTPENDING	R	master pending Indicates that the master is waiting to continue communication on the I2C-bus (pending) or is idle. When the master is pending, the MST STATE bits indicate what type of software service, if any, the master expects. This flag causes an interrupt when set, if enabled via the INT ENSET 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 required, mask this interrupt.	1
			0b in progress Communication is in progress. The master function is busy and cannot currently accept a command.	
			1b pending The master function requires 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.	

45.5.6.2.3 Interrupt enable set and read register (INTENSET)

Table 865. Interrupt enable set and read register - Offset: 808h

Bit	Name	Access	Description and value	Reset value
31:26	reserved	-	reserved Read value is undefined, only zero must be written.	-
25	SCLTIMEOUTEN	R/W	SCL timeout interrupt enable	0
			0b disabled The SCL timeout interrupt is disabled.	
			1b enabled The SCL timeout interrupt is enabled.	
24	EVENTTIMEOUTEN	R/W	event time-out interrupt enable	0
			0b disabled The event timeout interrupt is disabled.	
			1b enabled The event timeout interrupt is enabled.	
23:20	reserved	-	reserved Read value is undefined, only zero must be written.	-
19	MONIDLEEN	R/W	monitor idle flag interrupt enable	0
			0b disabled The MonIdle interrupt is disabled.	
			1b enabled The MonIdle interrupt is enabled.	
18	reserved	-	reserved Read value is undefined, only zero must be written.	-
17	MONOVEN	R/W	monitor overrun interrupt enable	0
			0b disabled The MonOv interrupt is disabled.	
			1b enabled The MonOv interrupt is enabled.	
16	MONRDYEN	R/W	monitor data ready interrupt enable	0
			0b disabled The MonRdy interrupt is disabled.	
			1b enabled The MonRdy interrupt is enabled.	

Table 865. Interrupt enable set and read register - Offset: 808h...continued

Bit	Name	Access	Description and value		Reset value
15	SLVDESELEN	R/W	slave deselect interrupt enable		0
			0b	disabled The SlvDeSel interrupt is disabled.	
			1b	enabled The SlvDeSel interrupt is enabled.	
14:12	reserved	-	reserved Read value is undefined, only zero must be written.		-
11	SLVNOTSTREN	R/W	slave not stretching interrupt enable		0
			0b	disabled The SlvNotStr interrupt is disabled.	
			1b	enabled The SlvNotStr interrupt is enabled.	
10:9	reserved	-	reserved Read value is undefined, only zero must be written.		-
8	SLVPENDINGEN	R/W	slave pending interrupt enable		0
			0b	disabled The SlvPending interrupt is disabled.	
			1b	enabled The SlvPending interrupt is enabled.	
7	reserved	-	reserved Read value is undefined, only zero must be written.		-
6	MSTSTSTPERREN	R/W	master start/stop error interrupt enable		0
			0b	disabled The MstStStpErr interrupt is disabled.	
			1b	enabled The MstStStpErr interrupt is enabled.	
5	reserved	-	reserved Read value is undefined, only zero must be written.		-
4	MSTARBLOSSEN	R/W	master arbitration loss interrupt enable		0
			0b	disabled The MstArbLoss interrupt is disabled.	
			1b	enabled The MstArbLoss interrupt is enabled.	

Table 865. Interrupt enable set and read register - Offset: 808h...continued

Bit	Name	Access	Description and value	Reset value
3:1	reserved	-	reserved Read value is undefined, only zero must be written.	-
0	MSTPENDINGEN	R/W	master pending interrupt enable	0
			0b disabled The MstPending interrupt is disabled.	
			1b enabled The MstPending interrupt is enabled.	

45.5.6.2.4 Interrupt enable clear register (INTENCLR)

Table 866. Interrupt enable clear register - Offset: 80Ch

Bit	Name	Access	Description and value	Reset value
31:26	reserved	-	reserved Read value is undefined, only zero must be written.	-
25	SCLTIMEOUTCLR	W	SCL time-out interrupt clear	0
24	EVENTTIMEOUTCLR	W	event timeout interrupt clear	0
23:20	reserved	-	reserved Read value is undefined, only zero must be written.	-
19	MONIDLECLR	W	monitor idle interrupt clear	0
18	reserved	-	reserved Read value is undefined, only zero must be written.	-
17	MONOVCLR	W	monitor overrun interrupt clear	0
16	MONRDYCLR	W	monitor data ready interrupt clear	0
15	SLVDESELCLR	W	slave deselect interrupt clear	0
14:12	reserved	-	reserved Read value is undefined, only zero must be written.	-
11	SLVNOTSTRCLR	W	slave not stretching interrupt clear	0
10:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8	SLVPENDINGCLR	W	slave pending interrupt clear	0
7	reserved	-	reserved Read value is undefined, only zero must be written.	-
6	MSTSTSTPERRCLR	W	master start/stop error interrupt clear	0

Table 866. Interrupt enable clear register - Offset: 80Ch...continued

Bit	Name	Access	Description and value	Reset value
5	reserved	-	reserved Read value is undefined, only zero must be written.	-
4	MSTARBLOSSCLR	W	master arbitration loss interrupt clear	0
3:1	reserved	-	reserved Read value is undefined, only zero must be written.	-
0	MSTPENDINGCLR	W	master pending interrupt clear Writing 1 to this bit clears the corresponding bit in the INTENSET register if implemented.	0

45.5.6.2.5 Timeout value register (TIMEOUT)

Table 867. Timeout value register - Offset: 810h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15:4	TO	R/W	timeout time value Specifies the timeout 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 timeouts, write a new value to TIMEOUT, then reenables timeouts. 0x000 = a timeout occurs after 16 counts of the I2C function clock. 0x001 = a timeout occurs after 32 counts of the I2C function clock. 0xFFFF = a timeout occurs after 65,536 counts of the I2C function clock.	1
3:0	TOMIN	R/W	timeout time value; bottom four bits Hard-wired to 0xF. Gives a minimum time-out of 16 I2C function clocks and also a timeout resolution of 16 I2C function clocks.	1

45.5.6.2.6 Clock predivider for the entire I2C interface (CLKDIV)

Table 868. Clock predivider for the entire I2C interface - Offset: 814h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15:0	DIVVAL	R/W	This field controls how the Flexcomm clock (FCLK) is used by the I2C functions that require an internal clock to operate. 0x0000 = FCLK is used directly by the I2C. 0x0001 = FCLK is divided by 2 before use. 0x0002 = FCLK is divided by 3 before use. 0xFFFF = FCLK is divided by 65,536 before use.	0

45.5.6.2.7 Interrupt status register for master, slave, and monitor functions (INTSTAT)

Table 869. Interrupt status register for master, slave, and monitor functions - Offset: 818h

Bit	Name	Access	Description and value	Reset value
31:26	reserved	-	reserved Read value is undefined, only zero must be written.	-
25	SCLTIMEOUT	R	SCL time-out Interrupt flag	0
24	EVENTTIMEOUT	R	event timeout interrupt flag	0
23:20	reserved	-	reserved Read value is undefined, only zero must be written.	-
19	MONIDLE	R	monitor idle flag	0
18	reserved	-	reserved Read value is undefined, only zero must be written.	-
17	MONOV	R	monitor overflow flag	0
16	MONRDY	R	monitor ready	0
15	SLVDESEL	R	slave deselected flag	0
14:12	reserved	-	reserved Read value is undefined, only zero must be written.	-
11	SLVNOTSTR	R	slave not stretching status	1
10:9	reserved	-	reserved Read value is undefined, only zero must be written.	-
8	SLVPENDING	R	slave pending	0
7	reserved	-	reserved Read value is undefined, only zero must be written.	-
6	MSTSTSPERR	R	master start/stop error flag	0
5	reserved	-	reserved Read value is undefined, only zero must be written.	-
4	MSTARBLOSS	R	master arbitration loss flag	0

Table 869. Interrupt status register for master, slave, and monitor functions - Offset: 818h...continued

Bit	Name	Access	Description and value	Reset value
3:1	reserved	-	reserved Read value is undefined, only zero must be written.	-
0	MSTPENDING	R	master pending	1

45.5.6.2.8 Master control register (MSTCTL)

Table 870. Master control register - Offset: 820h

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved Read value is undefined, only zero must be written.	-
3	MSTDMA	R/W	master DMA enable Data operations of the I2C can be performed with DMA. Protocol type operations such as start, address, stop, and address match must always be done with software, typically via an interrupt. Address acknowledgement must also be done by software except when the I2C is configured to be HSCAPABLE (and address acknowledgement is handled entirely by hardware) or when automatic operation is enabled. When a DMA data transfer is complete, MSTDMA must be cleared before beginning the next operation, typically a start or stop. This bit is read/write. 0b disable No DMA requests are generated for master operation. 1b enable A DMA request is generated for I2C master data operations. When this I2C master is generating acknowledge bits in master receiver mode, the acknowledge is generated automatically.	0
2	MSTSTOP	W	master stop control This bit is write-only. 0b no effect 1b stop A stop is generated on the I2C bus at the next allowed time, preceded by a NACK to the slave if the master is receiving data from the slave (master receiver mode).	0
1	MSTSTART	W	master start control This bit is write-only. 0b no effect 1b start A Start is generated on the I2C bus at the next allowed time.	0

Table 870. Master control register - Offset: 820h...continued

Bit	Name	Access	Description and value	Reset value
0	MSTCONTINUE	W	master continue This bit is write-only.	0
			0b no effect	
			1b continue Informs the master function to continue to the next operation. Must done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation.	

45.5.6.2.9 Master timing configuration (MSTTIME)

Table 871. Master timing configuration - Offset: 824h

Bit	Name	Access	Description and value	Reset value
31:7	reserved	-	reserved Read value is undefined, only zero must be written.	-
6:4	MSTSCLHIGH	R/W	master SCL high time Specifies the minimum high time that is asserted by this master on SCL. Other masters in a multimaster system can shorten this time. This time corresponds to the parameter tHIGH in the I2C-bus specification. I2C-bus specification parameters tSU;STO and tHD;STA have the same values and are also controlled by MSTSCLHIGH.	1
			000b 2 clocks Minimum SCL high time is 2 clocks of the I2C clock predivider.	
			001b 3 clocks Minimum SCL high time is 3 clocks of the I2C clock predivider.	
			010b 4 clocks Minimum SCL high time is 4 clocks of the I2C clock predivider.	
			011b 5 clocks Minimum SCL high time is 5 clocks of the I2C clock predivider.	
			100b 6 clocks Minimum SCL high time is 6 clocks of the I2C clock predivider.	
			101b 7 clocks Minimum SCL high time is 7 clocks of the I2C clock predivider.	

Table 871. Master timing configuration - Offset: 824h...continued

Bit	Name	Access	Description and value		Reset value
6:4 (cont'd)	MSTSCLHIGH	R/W	110b	8 clocks Minimum SCL high time is 8 clocks of the I2C clock predivider.	1
			111b	9 clocks Minimum SCL high time is 9 clocks of the I2C clock predivider.	
3	reserved	-	reserved Read value is undefined, only zero must be written.		-
2:0	MSTSCLLOW	R/W	master SCL low time Specifies the minimum low time that is asserted by this master on SCL. Other devices on the bus (masters or slaves) can increase this time. This time corresponds to the parameter tLOW in the I2C-bus specification. I2C-bus specification parameters tBUF and tSU;STA have the same values and are also controlled by MSTSCLLOW.		1
			000b	2 clocks Minimum SCL low time is 2 clocks of the I2C clock predivider.	
			001b	3 clocks Minimum SCL low time is 3 clocks of the I2C clock predivider.	
			010b	4 clocks Minimum SCL low time is 4 clocks of the I2C clock predivider.	
			011b	5 clocks Minimum SCL low time is 5 clocks of the I2C clock predivider.	
			100b	6 clocks Minimum SCL low time is 6 clocks of the I2C clock predivider.	
			101b	7 clocks Minimum SCL low time is 7 clocks of the I2C clock predivider.	
			110b	8 clocks Minimum SCL low time is 8 clocks of the I2C clock predivider.	
			111b	9 clocks Minimum SCL low time is 9 clocks of the I2C clock predivider.	

45.5.6.2.10 Combined master receiver and transmitter data register (MSTDAT)

Table 872. Combined master receiver and transmitter data register - Offset: 828h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	DATA	R/W	master function data register - Read: Reads the most recently received data for the master function. - Write: Transmits data using the master function.	0

45.5.6.2.11 Slave control register (SLVCTL)

Table 873. Slave control register - Offset: 840h

Bit	Name	Access	Description and value	Reset value
31:10	reserved	-	reserved Read value is undefined, only zero must be written.	-
9	AUTOMATCH READ	R/W	When AUTOACK is set, this bit controls if it matches a read or write request on the next header with an address matching SLVADR0. Since DMA must be configured to match the transfer direction, the direction must be specified. This bit allows a direction to be chosen for the next operation.	0
			0b The expected next operation in automatic mode is an I2C write.	
			1b The expected next operation in automatic mode is an I2C read.	
8	AUTOACK	R/W	automatic acknowledge When this bit is set, it causes an I2C header which matches SLVADR0 and the direction set by AUTOMATCHREAD to be ACKed immediately. It 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
			0b 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).	
			1b auto acknowledge A header with matching SLVADR0 and matching direction as set by AUTOMATCHREAD is ACKed immediately, allowing the master to move on to the data bytes. If the address matches SLVADR0, but the direction does not match AUTOMATCHREAD, the behavior depends on the AUTONACK bit in the SLVADR0 register: if AUTONACK is set, it is nacked. Otherwise, if AUTONACK is clear, a SlvPending interrupt is generated.	
7:4	reserved	-	reserved Read value is undefined, only zero must be written.	-

Table 873. Slave control register - Offset: 840h...continued

Bit	Name	Access	Description and value	Reset value
3	SLVDMA	R/W	slave DMA enable	0
			0b disabled No DMA requests are issued for Slave mode operation.	
			1b enabled DMA requests are issued for I2C slave data transmission and reception.	
2	reserved	-	reserved Read value is undefined, only zero must be written.	-
1	SLVNACK	R/W	slave NACK	0
			0b no effect	
			1b NACK Causes the slave function to NACK the master when the slave is receiving data from the master (slave receiver mode).	
0	SLVCONTINUE	R/W	slave continue	0
			0b no effect	
			1b continue Informs the slave function to continue to the next operation, by clearing the SLVPENDING flag in the STAT register. It 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 must not be set unless SLVPENDING = 1.	

45.5.6.2.12 Combined slave receiver and transmitter data register (SLVDAT)

Table 874. Combined slave receiver and transmitter data register - Offset: 844h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:0	DATA	R/W	slave function data register - Read: Reads the most recently received data for the slave function. - Write: Transmits data using the slave function.	0

45.5.6.2.13 Slave address register (SLVADR0)

Table 875. Slave address register - Offset: 848h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15	AUTONACK	R/W	automatic NACK operation Used in conjunction with AUTOACK and AUTOMATCHREAD. Allows software to ignore I2C traffic while handling previous I2C data or other operations.	0
			0b normal operation Matching I2C addresses are not ignored.	
			1b automatic-only mode All incoming addresses are ignored (NACKed), unless AUTOACK is set, it matches SLVADRn, and AUTOMATCHREAD matches the direction.	
14:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:1	SLVADR	R/W	slave address 7-bit slave address that is compared to received addresses, if enabled.	0
0	SADISABLE	R/W	slave address n disable	1
			0b enabled Slave address n is enabled.	
			1b ignored Slave address n is ignored.	

45.5.6.2.14 Slave address register (SLVADR1 - SLVADR3)

Table 876. Offset

Register	Offset
SLVADR1	84Ch
SLVADR2	850h
SLVADR3	854h

Table 877. Slave address register

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:1	SLVADR	R/W	slave address 7-bit slave address that is compared to received addresses, if enabled.	0
0	SADISABLE	R/W	slave address n disable	1
			0b enabled Slave address n is enabled.	
			1b ignored Slave address n is ignored.	

45.5.6.2.15 Slave qualification for address 0 (SLVQUAL0)

Table 878. Slave qualification for address 0 - Offset: 858h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved Read value is undefined, only zero must be written.	-
7:1	SLVQUAL0	R/W	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 causes 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 $SLVADR0[7:1] \leq \text{received address} \leq SLVQUAL0[7:1]$).	0
0	QUALMODE	R/W	qualify mode for slave address 0	0
			0b mask The SLVQUAL0 field is used as a logical mask for matching address 0.	
			1b extend The SLVQUAL0 field is used to extend address 0 matching in a range of addresses.	

45.5.6.2.16 Monitor receiver data register (MONRXDAT)

Table 879. Monitor receiver data register - Offset: 880h

Bit	Name	Access	Description and value	Reset value
31:11	reserved	-	reserved Read value is undefined, only zero must be written.	-
10	MONNACK	R	monitor received NACK 0b acknowledged The data that is currently provided by the monitor function was acknowledged by at least one master or slave receiver. 1b not acknowledged The data that is currently provided by the monitor function was not acknowledged by any receiver.	0
9	MONRESTART	R	monitor received repeated start 0b no repeated start detected The monitor function has not detected a repeated start event on the I2C-bus. 1b repeated start detected The monitor function has detected a repeated start event on the I2C-bus.	0
8	MONSTART	R	monitor received start 0b no start detected The monitor function has not detected a start event on the I2C-bus. 1b start detected The monitor function has detected a start event on the I2 C-bus.	0
7:0	MONRXDAT	R	monitor function receiver data Reflects every data byte that passes on the I2C pins.	0

45.5.7 General-purpose I/O

45.5.7.1 GPIO memory map

The GPIO registers are located at base address: 4008_C000h.

The SECGPIO registers are located at base address: 400A_8000h.

Table 880. General purpose I/O registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
B0_0 - B0_22	R/W	0h - 1Fh	byte pin registers for all port GPIO pins	8		Section 45.5.7.2.1
W0_0 - W0_22	R/W	1000h - 107Ch	word pin registers for all port GPIO pins	32	0000_0000h	Section 45.5.7.2.2
DIR0	R/W	2000h	direction registers for all port GPIO pins	32	0000-0000h	Section 45.5.7.2.3
MASK0	R/W	2080h	mask register for all port GPIO pins	32	0000_0000h	Section 45.5.7.2.4
PIN0	R/W	2100h	port pin register for all port GPIO pins	32	0000_0000h	Section 45.5.7.2.5
MPIN0	R/W	2180h	masked port register for all port GPIO pins	32	0000_0000h	Section 45.5.7.2.6
SET0	R/W	2200h	write: set register for port read: output bits for port	32	0000_0000h	Section 45.5.7.2.7
CLR0	W	2280h	clear port for all port GPIO pins	32		Section 45.5.7.2.8
NOT0	W	2300h	toggle port for all port GPIO pins	32		Section 45.5.7.2.9
DIRSET0	W	2380h	set pin direction bits for port	32		Section 45.5.7.2.10
DIRCLR0	W	2400h	clear pin direction bits for port	32		Section 45.5.7.2.11
DIRNOT0	W	2480h	toggle pin direction bits for port	32		Section 45.5.7.2.12

45.5.7.2 Register descriptions

45.5.7.2.1 Byte pin registers for all port GPIO pins (B0_n)

For n = 0 to 22

Table 881. Byte pin registers for all port GPIO pins - Offset: 0h + (n × 1h)

Bit	Name	Access	Description and value	Reset value
7:1	reserved	-	reserved 0 on read, ignored on write.	-
0	PBYTE	R/W	- 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. One register for each port pin. Supported pins depend on the specific device and package. - write: loads the output bit of the pin. One register byte for each port pin. Supported pins depend on the specific device and package.	0

45.5.7.2.2 Word pin registers for all port GPIO pins (W0_n)

For n = 0 to 22

Table 882. Word pin registers for all port GPIO pins - Offset: 1000h + (n × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	PWORD	R/W	- read 0: pin PIO0_n is LOW - write 0: clear output bit - read 0xFFFF FFFF: pin PIO0_n is HIGH - write any value between 0x0000 0001 and 0xFFFF FFFF: set output bit Only 0 or 0xFFFF FFFF can be read. Writing any value other than 0 sets the output bit. One register for each port pin. Supported pins depend on the specific device and package.	0

45.5.7.2.3 Direction registers for all port GPIO pins (DIR0)

Table 883. Direction registers for all port GPIO pins - Offset: 2000h

Bit	Name	Access	Description and value		Reset value
31:23	rerserved	-	reserved		-
22:0	DIRP	R/W	selects pin direction for pin PIO0_n (bit 0 = PIO0_0, bit 1 = PIO0_1, and so on) Supported pins depend on the specific device and package.		0
			0	input	
			1	output	

45.5.7.2.4 Mask register for all port GPIO pins (MASK0)

Table 884. Mask register for all port GPIO pins - Offset: 2080h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved	-
22:0	MASKP	R/W	controls which bits corresponding to PIO0_n are active in the MPORT register (bit 0 = PIO0_0, bit 1 = PIO0_1, and so on) Supported pins depend on the specific device and package.	0
			0 read MPORT: pin state; write MPORT: load output bit	
			1 read MPORT: 0; write MPORT: output bit not affected	

45.5.7.2.5 Port pin register for all port GPIO pins (PIN0)

Table 885. Port pin register for all port GPIO pins - Offset: 2100h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved	-
22:0	PORT	R/W	Reads pin states or loads output bits (bit 0 = PIO0_0, bit 1 = PIO0_1, and so on) Supported pins depend on the specific device and package.	0
			0 read: pin is low; write: clear output bit	
			1 read: pin is high; write: set output bit	

45.5.7.2.6 Masked port register for all port GPIO pins (MPIN0)

Table 886. Masked port register for all port GPIO pins - Offset: 2180h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved	-
22:0	MPORTP	R/W	masked port register (bit 0 = PIO0_0, bit 1 = PIO0_1, and so on) Supported pins depend on the specific device and package.	0
			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.	

45.5.7.2.7 Write: Set register for port; Read: output bits for port (SET0)

Table 887. Write: Set register for port; Read: output bits for port - Offset: 2200h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved	-
22:0	SETP	R/W	read or set output bits (bit 0 = PIO0_0, bit 1 = PIO0_1, and so on) Supported pins depend on the specific device and package.	0
			0 read: output bit; write: no operation.	
			1 read: output bit; write: set output bit	

45.5.7.2.8 Clear port for all port GPIO pins (CLR0)

Table 888. Clear port for all port GPIO pins - Offset: 2280h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved	-
22:0	CLRP	W	clear output bits (bit 0 = PIO0_0, bit 1 = PIO0_1, and so on) Supported pins depend on the specific device and package.	0
			0 no operation.	
			1 clear output bit	

45.5.7.2.9 Toggle port for all port GPIO pins (NOT0)

Table 889. Toggle port for all port GPIO pins - Offset: 2300h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved	-
22:0	NOTP	W	toggle output bits (bit 0 = PIO0_0, bit 1 = PIO0_1, and so on) Supported pins depend on the specific device and package.	0
			0 no operation.	
			1 toggle output bit	

45.5.7.2.10 Set pin direction bits for port (DIRSET0)

Table 890. Set pin direction bits for port - Offset: 2380h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved	-
22:0	DIRSETP	W	set direction bits (bit 0 = PIO0_0, bit 1 = PIO0_1, and so on) Supported pins depend on the specific device and package.	0
			0 no operation.	
			1 set direction bit	

45.5.7.2.11 Clear pin direction bits for port (DIRCLR0)

Table 891. Clear pin direction bits for port - Offset: 2400h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved	-
22:0	DIRCLRP	W	clear direction bits (bit 0 = PIO0_0, bit 1 = PIO0_1, and so on) Supported pins depend on the specific device and package.	0
			0 no operation.	
			1 clear direction bit	

45.5.7.2.12 Toggle pin direction bits for port (DIRNOT0)

Table 892. Toggle pin direction bits for port - Offset: 2480h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved	-
22:0	DIRNOTP	W	toggle direction bits (bit 0 = PIO0_0, bit 1 = PIO0_1, and so on) Supported pins depend on the specific device and package.	0
			0 no operation.	
			1 toggle direction bit	

45.5.8 CRC engine (CRC)

45.5.8.1 CRC memory map

The CRC_ENGINE registers are located at base address: 4009_5000h.

Table 893. CRC engine registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
MODE	R/W	0h	CRC mode register	32		Section 45.5.8.2.1
SEED	R/W	4h	CRC seed register	32	0000_FFFFh	Section 45.5.8.2.2
SUM	R	8h	CRC checksum register	32	0000-FFFFh	Section 45.5.8.2.3
WR_DATA	W	8h	CRC data register	32		Section 45.5.8.2.4

45.5.8.2 Register descriptions

45.5.8.2.1 CRC mode register (MODE)

Table 894. CRC mode register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved Always 0 when read.	-
5	CMPL_SUM	R/W	CRC sum complement	0
			1 1's complement for CRC_SUM	
			0 no 1's complement for CRC_SUM	
4	BIT_RVS_SUM	R/W	CRC sum bit order	0
			1 bit order reverse for CRC_SUM	
			no bit order reverse for CRC_SUM	
3	CMPL_WR	R/W	data complement	0
			1 1's complement for CRC_WR_DATA	
			0 no 1's complement for CRC_WR_DATA	
2	BIT_RVS_WR	R/W	data bit order	0
			1 bit order reverse for CRC_WR_DATA (per byte)	
			0 no bit order reverse for CRC_WR_DATA (per byte)	
1:0	CRC_POLY	R/W	CRC polynomial	0
			1X CRC-32 polynomial	
			01 CRC-16 polynomial	
			00 CRC-CCITT polynomial	

45.5.8.2.2 CRC seed register (SEED)

Table 895. CRC seed register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:0	CRC_SEED	R/W	A write access to this register loads a CRC seed value to CRC_SUM register with the selected bit order and 1's complement preprocesses. A write access to this register overrules the CRC calculation in progress.	0

45.5.8.2.3 CRC checksum register (SUM)

Table 896. CRC checksum register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:0	CRC_SUM	R	The most recent CRC sum can be read through this register with selected bit order and 1's complement postprocesses.	0

45.5.8.2.4 CRC data register (WR_DATA)

Table 897. CRC data register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:0	CRC_WR_DATA	W	Data written to this register are taken to perform a CRC calculation with the selected bit order and the 1's complement preprocess. Any write size 8-bit, 16-bit, or 32-bit are allowed and accept back-to-back transactions.	0

45.5.9 Debug mailbox (DBG)

45.5.9.1 MCU debugger mailbox memory map

The MCU debugger mailbox registers are located at base address: 4009_C000h.

Table 898. MCU debugger mailbox engine registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CSW	R/W	0h	debug mailbox control and status	32		Section 45.5.9.2.1
REQUEST	R/W	4h	debug mailbox request	32	0000_FFFFh	Section 45.5.9.2.2
RETURN	R/W	8h	debug mailbox return value	32	0000-0000h	Section 45.5.9.2.3
ID	r	FCh	identification register	32	002A_0000h	Section 45.5.9.2.4

45.5.9.2 Register descriptions

45.5.9.2.1 Debug mailbox control and status register (CSW)

Table 899. Debug mailbox control and status register - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	Always 0 when read	-
5	CHIP_RESET_REQ	W	When written, causes the chip to reset Note: The DM is not reset by this reset as it is only resettable by a SOFT reset or a POR/BOD event)	0
4	SOFT_RESET	R/W	soft reset for DM (write-only from AHB; not readable and self-clearing) A write to this bit causes a soft reset for DM.	0
3	AHB_OR_ERR	R/W	AHB overrun error Return value overwritten by ROM.	0
2	DBG_OR_ERR	R/W	debugger overrun error Previous REQUEST is overwritten before it is picked up by ROM.	0
1	REQ_PENDING	R/W	request is pending from debugger (that is, unread value in REQUEST)	0
0	RESYNCH_REQ	R/W	debugger sets this bit to 1 to request a resynchronisation	0

45.5.9.2.2 Debug mailbox request (REQUEST)

Table 900. Debug mailbox request register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:0	REQ	R/W	request value Read as 0x0 when no new request is present. Note: Only an AHB side read clears the request. The debugger is allowed to read back the request register, if it wants to confirm that the communication is established (that is, that the MD is clocking correctly).	0

45.5.9.2.3 Debug mailbox return value (RETURN)

Table 901. Debug mailbox return value register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:0	RET	R/W	return value Any response from the device to the debugger. If no new data is present, a debugger read is stalled until new data is available.	0

45.5.9.2.4 Identification register (ID)

Table 902. Identification register - Offset: FCh

Bit	Name	Access	Description and value	Reset value
31:0	ID	R	identification value	0

45.5.10 SPIFI slave port (SPIFI)

45.5.10.1 SPIFI memory map

The SPIFI registers are located at base address: 4009_F000h.

Table 903. SPI flash interface registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CTRL	R/W	0h	SPIFI control register	32		Section 45.5.10.2.1
CMD	R/W	4h	SPIFI command register	32	0000_0000h	Section 45.5.10.2.2
ADDR	R/W	8h	SPIFI address register	32	0000_0000h	Section 45.5.10.2.3
IDATA	R/W	Ch	SPIFI intermediate data register	32	0000_0000h	Section 45.5.10.2.4
CLIMIT	R/W	10h	SPIFI limit register	32	0800_0000h	Section 45.5.10.2.5
DATA	R/W	14h	SPIFI data register	32	0000_0000h	Section 45.5.10.2.6
MCMD	R/W	18h	SPIFI memory command register	32		Section 45.5.10.2.7
STAT	R/W	1Ch	SPIFI status register	32		Section 45.5.10.2.8

45.5.10.2 Register descriptions

45.5.10.2.1 SPIFI control register (CTRL)

Table 904. SPIFI control register - Offset: 0h

Bit	Name	Access	Description and value		Reset value
31	DMAEN	R/W	A 1 in this bit enables the DMA request output from the SPIFI. Only set this bit when a DMA channel is used to transfer data in peripheral mode. Do not set this bit when a DMA channel is used for memory-to-memory transfers from the SPIFI memory area. DMAEN must only be used in command mode.		0
30	FBCLK	R/W	feedback clock select		1
			0b	internal clock The SPIFI samples read data using an internal clock.	
			1b	feedback clock Read data is sampled using a feedback clock from the SCK pin. It allows slightly more time for each received bit. MODE3, RFCLK, and FBCLK must not all be 1, because, in that case, there is no final falling edge on SCK on which to sample the last data bit of the frame.	
29	RFCLK	R/W	select active clock edge for input data		0
			0b	rising edge Read data is sampled on rising edges on the clock, as in classic SPI operation.	
			1b	falling edge Read data is sampled on falling edges of the clock, allowing a full serial clock of time to maximize the serial clock frequency. MODE3, RFCLK, and FBCLK must not all be 1, because, in that case, there is no final falling edge on SCK on which to sample the last data bit of the frame.	
28	DUAL	R/W	select dual protocol		0
			0b	quad protocol This protocol uses IO3:0.	
			1b	dual protocol This protocol uses IO1:0.	
27	PRFTCH_DIS	R/W	cache prefetching enable The SPIFI includes an internal cache. A 1 in this bit disables prefetching of cache lines.		0
			0b	enable Cache prefetching enabled.	
			1b	disable Disables prefetching of cache lines.	
26:24	reserved	-	reserved		-

Table 904. SPIFI control register - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value
23	MODE3	R/W	SPI mode 3 select	0
			0b SCK LOW The SPIFI drives SCK low after the rising edge. The last bit of each command is captured and keeps it LOW while CS is HIGH.	
			1b SCK HIGH The SPIFI keeps SCK high after the rising edge for the last bit of each command and while CS is HIGH, and drives it low after it drives CS LOW (Known serial flash devices can handle either mode, but some devices may require a particular mode for proper operation). MODE3, RFCLK, and FBCLK must not all be 1, because, in that case, there is no final falling edge on SCK on which to sample the last data bit of the frame.	
22	INTEN	R/W	If this bit is 1 when a command ends, the SPIFI asserts its interrupt request output. See INTRQ in Table 911 for further details.	0
21	D_PRFTCH_DIS	R/W	This bit allows conditioning of memory mode prefetches based on the AHB HPROT (instruction/data) access information. A 1 in this register means that the SPIFI does not attempt a speculative prefetch when it encounters data accesses.	0
20	reserved	-	reserved	-
19:16	CSHIGH	R/W	This field controls the minimum CS high time, expressed as a number of serial clock periods minus one.	1
15:0	TIMEOUT	R/W	This field contains the number of serial clock periods without the processor reading data in memory mode, which causes the SPIFI hardware to terminate the command by driving the CS pin high and negating the CMD bit in the status register (allows the flash memory to enter a lower-power state). If the processor reads data from the flash region after a timeout, the command in the memory command register is issued again.	1

45.5.10.2.2 SPIFI command register (CMD)

Table 905. SPIFI command register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:24	OPCODE	R/W	opcode of the command (not used for some FRAMEFORM values)	0
23:21	FRAMEFORM	R/W	controls the opcode and address fields	0
			000b reserved	
			001b opcode Opcode only, no address.	
			010b opcode one byte Opcode, least significant byte of address.	
			011b opcode two bytes Opcode, two least significant bytes of address.	
			100b opcode three bytes Opcode, three least significant bytes of address.	
			101b opcode four bytes Opcode, 4 bytes of address.	
			110b no opcode three bytes No opcode, 3 least significant bytes of address.	
			111b no opcode four bytes No opcode, 4 bytes of address.	
20:19	FIELDFORM	R/W	controls how the fields of the command are sent	0
			00b all serial All fields of the command are serial.	
			01b quad/dual data Data field is quad/dual, other fields are serial.	
			10b serial opcode Opcode field is serial. Other fields are quad/dual.	
			11b all quad/dual All fields of the command are in quad/dual format.	
18:16	INTLEN	R/W	This field controls how many intermediate bytes precede the data (each byte may require 8 SCK cycles or 2 SCK cycles, depending on if the intermediate field is in serial, 2-bit, or 4-bit format). Intermediate bytes are output by the SPIFI, and include post-address control information, dummy and delay bytes (see the description of the intermediate data register for the contents of such bytes).	0
15	DOUT	R/W	If the DATALEN field is not zero, this bit controls the direction of the data.	0
			0b input from serial flash	
			1b output to serial flash	

Table 905. SPIFI command register - Offset: 4h...continued

Bit	Name	Access	Description and value	Reset value
14	POLL	R/W	This bit must be written as 1 only with an opcode that: - Contains an input data field - Causes the serial flash device to return byte status repetitively (for example, a read status command) When this bit is 1, the SPIFI hardware continues to read bytes until the test specified by the DATALEN field is met. The hardware tests the bit in each status byte selected by DATALEN bits 2:0, until a bit is found that equals DATALEN bit 3. When the test succeeds, the SPIFI captures the byte that meets this test so that it can be read from the data register and terminates the command by raising CS. The end-of-command interrupt can be enabled to inform software when this occurs.	0
13:0	DATALEN	R/W	Except when the POLL bit in this register is 1, this field controls how many data bytes are in the command. 0 indicates that the command does not contain a data field.	0

45.5.10.2.3 SPIFI address register (ADDR)

Table 906. SPIFI address register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:0	ADDR	R/W	address	0

45.5.10.2.4 SPIFI intermediate data register (IDATA)

Table 907. SPIFI intermediate data register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:0	IDATA	R/W	value of intermediate bytes	0

45.5.10.2.5 SPIFI limit register (CLIMIT)

Table 908. SPIFI limit register - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:0	CLIMIT	R/W	zero-based upper limit of cacheable memory	0

45.5.10.2.6 SPIFI data register (DATA)

Table 909. SPIFI data register - Offset: 14h

Bit	Name	Access	Description and value	Reset value
31:0	DATA	R/W	input or output data	0

45.5.10.2.7 SPIFI memory command register (MCMD)

Table 910. SPIFI memory command register - Offset: 18h

Bit	Name	Access	Description and value	Reset value
31:24	OPCODE	R/W	opcode of the command (not used for some FRAMEFORM values)	0
23:21	FRAMEFORM	R/W	controls the opcode and address fields	0
			000b reserved	
			001b opcode Opcode only, no address.	
			010b opcode one byte Opcode, least significant byte of address.	
			011b opcode two bytes Opcode, two least significant bytes of address.	
			100b opcode three bytes Opcode, three least significant bytes of address.	
			101b opcode four bytes Opcode, 4 bytes of address.	
			110b no opcode three bytes No opcode, 3 least significant bytes of address.	
			111b no opcode four bytes No opcode, 4 bytes of address.	
20:19	FIELDFORM	R/W	controls how the fields of the command are sent	0
			00b all serial All fields of the command are serial.	
			01b quad/dual data Data field is quad/dual, other fields are serial.	
			10b serial opcode Opcode field is serial. Other fields are quad/dual.	
			11b all quad/dual All fields of the command are in quad/dual format.	
18:16	INTLEN	R/W	This field controls how many intermediate bytes precede the data (each byte may require 8 SCK cycles or 2 SCK cycles, depending on if the intermediate field is in serial, 2-bit, or 4-bit format). Intermediate bytes are output by the SPIFI, and include post-address control information, dummy and delay bytes (see the description of the intermediate data register for the contents of such bytes).	0
15	DOUT	R/W	This bit must be written as 0.	0
14	POLL	R/W	This bit must be written as 0.	0
13:0	reserved	-	reserved	-

45.5.10.2.8 SPIFI status register (STAT)

Table 911. SPIFI status register - Offset: 1Ch

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved	-
5	INTRQ	R/W	reflects the SPIFI interrupt request Write a 1 to this bit to clear it. This bit is set when a CMD was previously 1 and has been cleared due to the deassertion of CS.	0
4	RESET	R/W	Write a 1 to this bit to abort a current command or memory mode. This bit is cleared when the hardware is ready for a new command to be written to the command register.	0
3:2	reserved	-	reserved	-
1	CMD	R/W	1 when the command register is written. This bit is cleared by a hardware reset, a write to the RESET bit in this register, or the deassertion of CS which indicates that the command has completed communication with the SPI flash.	0
0	MCINIT	R/W	Set when software successfully writes the memory command register. This bit is cleared by reset or by writing a 1 to the RESET bit in this register.	0

45.5.11 Code watchdog (CDOG)

45.5.11.1 CDOG memory map

The code watchdog registers are located at base address: 400A_1000h.

Table 912. Code watchdog registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CONTROL	R/W	0h	code watchdog control	32	5009_2492h	Section 45.5.11.2.1
RELOAD	R/W	4h	instruction timer reload	32	FFFF_FFFFh	Section 45.5.11.2.2
INSTRUCTION_TIMER	R/W	8h	instruction timer	32	FFFF_FFFFh	Section 45.5.11.2.3
SECURE_COUNTER	R/W	Ch	secure counter	32	0000_0000h	Section 45.5.11.2.4
STATUS	R	10h	status register 1	32	0000_0000h	Section 45.5.11.2.5
STATUS2	R	14h	status register 2	32	0000_0000h	Section 45.5.11.2.6
FLAGS	R/W	18h	hardware flags	32	0000_0000h	Section 45.5.11.2.7
PERSISTENT	R/W	1Ch	persistent data storage	32	0000_0000h	Section 45.5.11.2.8
START	W	20h	write address for issuing the START command	32	0000_0000h	Section 45.5.11.2.9
STOP	W	24h	write address for issuing the STOP command	32	0000_0000h	Section 45.5.11.2.10
RESTART	W	28h	write address for issuing the RESTART command	32	0000_0000h	Section 45.5.11.2.11
ADD	W	2CVh	write address for issuing the ADD command	32	0000_0000h	Section 45.5.11.2.12
ADD1	W	30h	write address for issuing the ADD1 command	32	0000_0000h	Section 45.5.11.2.13
ADD16	W	340h	write address for issuing the ADD16 command	32	0000_0000h	Section 45.5.11.2.14
ADD256	W	38h	write address for issuing the ADD256 command	32	0000_0000h	Section 45.5.11.2.15
SUB	W	3Ch	write address for issuing the SUB command	32	0000_0000h	Section 45.5.11.2.16
SUB1	W	40h	write address for issuing the SUB1 command	32	0000_0000h	Section 45.5.11.2.17
SUB16	W	44h	write address for issuing the SUB16 command	32	0000_0000h	Section 45.5.11.2.18
SUB256	W	48h	write address for issuing the SUB256 command	32	0000_0000h	Section 45.5.11.2.19
ASSERT16	R/W	4Ch	write address for issuing the ASSERT16 command	32	0000_0000h	Section 45.5.11.2.20

45.5.11.2 Register descriptions

45.5.11.2.1 Code watchdog control (CONTROL)

Houses the bit-fields that exercise control over the various behavioral aspects of the module. Writable and readable only when LOCK_CTRL = unlocked.

Table 913. Code watchdog control - Offset: 0h

Bit	Name	Access	Description and value		Reset value
31:30	DEBUG_HALT_CTRL	R/W	DEBUG_HALT control field		1
			01b	run	
			10b	pause the instruction timer when CPU asserts debug HALT. Reset value is run.	
29:28	IRQ_PAUSE	R/W	IRQ pause control field		4
			01b	instruction timer runs during interrupts	
			10b	instruction timer pauses during interrupts Reset value is run.	
27:20	reserved	-	reserved		-
19:17	ADDRESS_CTRL	R/W	ADDRESS control field		1
			001	enable reset	
			010	enable interrupt	
			100	disable both when address fault occurs Resets to disable_both.	
16:14	STATE_CTRL	R/W	1 when the command register is written. This bit is cleared by a hardware reset, a write to the RESET bit in this register, or the deassertion of CS which indicates that the command has completed communication with the SPI flash.		4
			001	enable reset	
			010	enable interrupt	
			100	disable both when illegal state fault occurs Resets to disable_both.	
13:11	CONTROL_CTRL	R/W	CONTROL control field		4
			001	enable reset	
			010	enable interrupt	
			100	disable both when illegal CONTROL fault occurs Resets to disable_both.	

Table 913. Code watchdog control - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value
10:8	SEQUENCE_CTRL	R/W	SEQUENCE control field	4
			001 enable reset	
			010 enable interrupt	
			100 disable both when illegal sequence fault occurs Resets to disable_both.	
7:5	MISCOMPARE_CTRL	R/W	MISCOMPARE control field	4
			001 enable reset	
			010 enable interrupt	
			100 disable both when miscompare fault occurs Resets to disable_both.	
4:2	TIMEOUT_CTRL	R/W	TIMEOUT control	4
			001 enable reset	
			010 enable interrupt	
			100 disable both when timeout occurs Resets to disable_both.	
1:0	LOCK_CTRL	R/W	Lock control field	2
			01b locked	
			10b unlocked Resets to unlocked.	

45.5.11.2.2 Instruction timer reload (RELOAD)

Contains the value from which the INSTRUCTION TIMER counts down when a valid START or RESTART command is executed. Can only be written to when the state machine is in IDLE state. Must be written before issuing a START command.

Table 914. Instruction timer reload - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:0	RLOAD	R/W	instruction timer reload value	0xFFFFFFFF

45.5.11.2.3 Instruction timer (INSTRUCTION_TIMER)

The current value of the INSTRUCTION TIMER can be read from this address at all times. Not writable by the CPU.

Table 915. Instruction timer - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:0	INSTIM	R/W	INSTRUCTION TIMER 32-bit value	0xFFFFFFFF

45.5.11.2.4 Secure counter (SECURE_COUNTER)

The SECURE_COUNTER is a write-only-register. Write access is only available indirectly, via writes issued to the command register group.

Table 916. Secure counter - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:0	SECCNT	W	secure counter	0

45.5.11.2.5 Status register 1 (STATUS)

Current status of the CDOG is available here for inspection. The fields in this register are reset only by POR.

Table 917. Status register 1 - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:28	reserved	-	reserved	-
27:24	CURST	R	current state of the STATE MACHINE Resets to 0x5 = IDLE.	0
23:16	NUMILSEQF	R	number of illegal sequence faults since the last POR Resets to 0. Does not increment past 0xFF.	0
15:8	NUMMISCOMPF	R	number of miscompare faults since the last POR Resets to 0. Does not increment past 0xFF.	0
7:0	NUMTOF	R	number of timeout faults since the last POR Resets to 0. Does not increment past 0xFF.	0

45.5.11.2.6 Status register 2 (STATUS2)

Additional status of the code watchdog is available here for introspection. The fields in this register are reset only by POR.

Table 918. Status register 2 - Offset: 14h

Bit	Name	Access	Description and value	Reset value
31:24	reserved	-	reserved	-
23:16	NUMILLA	R	number of illegal address faults since the last POR Resets to 0. Does not increment past 0xFF.	0
15:8	NUMILLSTF	R	number of illegal state faults since the last POR Resets to 0. Does not increment past 0xFF.	0
7:0	NUMCNTF	R	number of illegal control faults since the last POR Resets to 0. Does not increment past 0xFF.	0

45.5.11.2.7 Hardware flags (FLAGS)

The hardware flag bits (themselves) can be found here. When the code watchdog is enabled in the final application, the flags may only be set by hardware, and only be cleared by software or power-on reset. During code development, the flags may be set by software to generate faults.

Table 919. Hardware flags - Offset: 18h

Bit	Name	Access	Description and value	Reset value
31:17	reserved	-	reserved	-
16	POR_FLAG	R/W	power-on reset flag When unlocked, write-direct. When locked, WR-1-to-clear. Always set by POR. Used to determine if the previous reset was a power-on reset or some other reset flavor.	0
15:6	reserved	-	reserved	-
5	ADDR_FLAG	R/W	address flag When unlocked, write-direct. When locked, WR-1-to-clear. Set by hardware upon detection of an illegal address_fault. When '1', this bit causes a reset or an interrupt if enabled to do so (see Section 45.5.11.2.1).	0
4	STATE_FLAG	R/W	state flag When unlocked, write-direct. When locked, WR-1-to-clear. Set by hardware upon detection of an illegal state_fault. When '1', this bit causes a reset or an interrupt if enabled to do so (see Section 45.5.11.2.1).	0
3	CNT_FLAG	R/W	control (fault) flag When unlocked, write-direct. When locked, WR-1-to-clear. Set by hardware upon detection of an illegal control_fault. When '1', this bit causes a reset or an interrupt if enabled to do so (see Section 45.5.11.2.1).	0
2	SEQ_FLAG	R/W	sequence flag When unlocked, write-direct. When locked, WR-1-to-clear. Set by hardware upon detection of an illegal sequence_fault. When '1', this bit causes a reset or an interrupt if enabled to do so (see Section 45.5.11.2.1).	0
1	MISCOM_FLAG	R/W	miscompare flag When unlocked, write-direct. When locked, WR-1-to-clear. Set by hardware upon detection of an illegal miscompare fault. When '1', this bit causes a reset or an interrupt if enabled to do so (see Section 45.5.11.2.1).	0
0	TO_FLAG	R/W	timeout When unlocked, write-direct. When locked, WR-1-to-clear. Set by hardware upon detection of an illegal timeout_ault. When '1', this bit causes a reset or an interrupt if enabled to do so (see Section 45.5.11.2.1).	0

45.5.11.2.8 Persistent data storage (PERSISTENT)

Software can store 32 bits of data here for retrieval after ANY reset other than POR.

Table 920. Persistent data storage - Offset: 1Ch

Bit	Name	Access	Description and value	Reset value
31:0	PERSIS	R/W	32 regs free for user	0

45.5.11.2.9 Write address for issuing the START command (START)

Writing to this address loads the secure_counter with the 32-bits of write data and starts the instruction timer. Can only be written during IDLE state.

Table 921. Write address for issuing the START command - Offset: 20h

Bit	Name	Access	Description and value	Reset value
31:0	STRT	W	address of start command access	0

45.5.11.2.10 Write address for issuing the STOP command (STOP)

Writing to this address stops the instruction timer and forces a compare between the secure counter and the 32 bits of write data. Can only be written during ACTIVE state.

Table 922. Write address for issuing the STOP command - Offset: 24h

Bit	Name	Access	Description and value	Reset value
31:0	STP	W	address of stop command access	0

45.5.11.2.11 Write address for issuing the RESTART command (RESTART)

Writing to this address forces a compare between the secure counter and the 32 bits of write data. If the values are identical, the instruction timer is reloaded and starts downcounting again. Can only be written during ACTIVE state.

Table 923. Write address for issuing the RESTART command - Offset: 28h

Bit	Name	Access	Description and value	Reset value
31:0	RSTRT	W	write address for issuing the RESTART command	0

45.5.11.2.12 Write address for issuing the ADD command (ADD)

The value written to this address is added to the current value of the SECURE COUNTER. Can only be written during ACTIVE state.

Table 924. Write address for issuing the ADD command - Offset: 2Ch

Bit	Name	Access	Description and value	Reset value
31:0	AD	W	address of the ADD command	0

45.5.11.2.13 Write address for issuing the ADD1 command (ADD1)

The value written to this address is ignored. The value '1' is added to the current value of the SECURE COUNTER. Can only be written during ACTIVE state.

Table 925. Write address for issuing the ADD1 command - Offset: 30h

Bit	Name	Access	Description and value	Reset value
31:0	AD1	W	address of the ADD1 command	0

45.5.11.2.14 Write address for issuing the ADD16 command (ADD16)

The value written to this address is ignored. The value 16d is added to the current value of the SECURE COUNTER. Can only be written during ACTIVE state.

Table 926. Write address for issuing the ADD16 command - Offset: 34h

Bit	Name	Access	Description and value	Reset value
31:0	AD16	W	address of the ADD16 command	0

45.5.11.2.15 Write address for issuing the ADD256 command (ADD256)

The value written to this address is ignored, and the value 256d is added to the current value of the SECURE COUNTER. Can only be written during ACTIVE state.

Table 927. Write address for issuing the ADD256 command - Offset: 38h

Bit	Name	Access	Description and value	Reset value
31:0	AD256	W	address of the ADD256 command	0

45.5.11.2.16 Write address for issuing the SUB command (SUB)

The value written to this address is subtracted from the current value of the SECURE COUNTER. Can only be written during ACTIVE state.

Table 928. Write address for issuing the SUB command - Offset: 3Ch

Bit	Name	Access	Description and value	Reset value
31:0	S0B	W	address of the SUB command	0

45.5.11.2.17 Write address for issuing the SUB1 command (SUB1)

The value written to this address is ignored. The value '1' is subtracted from the current value of the SECURE COUNTER. Can only be written during ACTIVE state.

Table 929. Write address for issuing the SUB1 command - Offset: 40h

Bit	Name	Access	Description and value	Reset value
31:0	S1B	W	address of the SUB1 command	0

45.5.11.2.18 Write address for issuing the SUB16 command (SUB16)

The value written to this address is ignored. The value 16d is subtracted from the current value of the SECURE COUNTER. Can only be written during ACTIVE state.

Table 930. Write address for issuing the SUB16 command - Offset: 44h

Bit	Name	Access	Description and value	Reset value
31:0	SB16	W	address of the SUB16 command	0

45.5.11.2.19 Write address for issuing the SUB256 command (SUB16)

The value written to this address is ignored. The value 256d is subtracted from the current value of the SECURE COUNTER. Can only be written during ACTIVE state.

Table 931. Write address for issuing the SUB256 command - Offset: 48h

Bit	Name	Access	Description and value	Reset value
31:0	SB256	W	address of the SUB256 command	0

45.5.11.2.20 Write address for issuing the ASSERT16 command (ASSERT16)

The low 16-bit value written to the ASSERT16 command register is compared with the current low 16-bit value of the secure counter. A MISCOMPARE fault occurs when the two values compared are not equal.

Table 932. Write address for issuing the ASSERT16 command - Offset: 4Ch

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved Read value is undefined, only zero must be written.	-
15:0	ASSERT16	W	assert value	

45.5.12 Hash-AES peripheral (HASHAES)

45.5.12.1 HASHAES memory map

The HASHAES registers are located at base address: 400A_4000h.

Table 933. HASHAES peripheral registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CTRL	R/W	0h	hash control register	32		Section 45.5.12.2.1
STATUS	W	4h	hash status register	32		Section 45.5.12.2.2
INTENSET	R/W	8h	hash interrupt enable set register	32		Section 45.5.12.2.3
INTENCLR	W	Ch	hash interrupt enable clear register	32		Section 45.5.12.2.4
MEMCTRL	R/W	10h	memory access control	32		Section 45.5.12.2.5
MEMADDR	R/W	14h	memory access start address	32	0000_0000h	Section 45.5.12.2.6
INDATA	W	20h	16 words input data	32		Section 45.5.12.2.7
ALIAS0 - ALIAS6	W	24h - 3Ch	aliases for input data	32		Section 45.5.12.2.8
DIGEST00 - DIGEST07	R	40h - 5Ch	result digest	32	0000_0000h	Section 45.5.12.2.9
DIGEST10 - DIGEST17	R	60h - 7Ch	digest result, part 2	32	0000_0000h	Section 45.5.12.2.10
CRYPTCFG	R/W	80h	crypto configuration	32		Section 45.5.12.2.11
CONFIG	W	84h	hardware configuration	32		Section 45.5.12.2.12
LOCK	R/W	8Ch	register lock	32		Section 45.5.12.2.13
MASK0 - MASK3	W	90h - 9Ch	ICB mask	32		Section 45.5.12.2.14
RELOAD0 - RELOAD7	W	A0h - BCh	SHA digest to reload	32		Section 45.5.12.2.15
PRNG_SEED	W	D0h	pseudo random number generator seed value	32	0000_0000h	Section 45.5.12.2.16
PRNG_OUT	R	D8h	pseudo random number	32		Section 45.5.12.2.17

45.5.12.2 Register descriptions

45.5.12.2.1 Hash control register (CTRL)

Table 934. Hash control register (CTRL) - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:14	reserved	-	reserved	-
13	AESFLUSH	W	flushes the AES engine registers This bit self clears.	0
			0b do not flush the AES engine registers	
			1b flush the AES engine registers	
12	HASHSWPB	R/W	if 1, swaps bytes in the word for SHA hashing The default is byte order (LSB is 1st byte). However, swapping to MSB is 1st such as is shown in the SHS specification, is allowed. For cryptographic swapping, see Section 45.5.12.2.11 .	0
11:10	reserved	-	reserved	-
9	DMA_O	R/W	written to 1 to use DMA to drain the digest/output If DMA_I and DMA_O are set, the DMA must know how to switch direction and the locations. Can be used for crypto.	0
			0b DMA not used Processor reads the digest/output in response to DIGEST interrupt.	
			1b DMA enabled	
8	DMA_I	R/W	written with 1 to use DMA to fill INDATA If hash, requests 16 words from DMA and then processes the hash. If cryptographic, it loads as many words as required, including the key if it is not already loaded. It then requests again. Normal model is that the DMA interrupts the processor when its length expires. Note that Note: <i>If the processor writes the key and optionally IV, it must not enable it until it has finished done so. Otherwise, the DMA is expected to load those for the 1st block (when necessary).</i>	0
			0b DMA not used The processor writes the necessary words when WAITING is set (interrupts), unless AHB master is used.	
			1b DMA pushes in the data	
7:6	reserved	-	reserved	-
5	RELOAD	R/W	If 1, allows the SHA RELOAD registers to be used Used to save a partial hash digest (for example, when running AES is required) and reload it later for continuation.	0
4	NEW_HASH	W	Written with 1 when starting a new hash/crypto This bit self clears. Note: <i>The WAITING status bit clears for a cycle during the initialization from New = 1.</i>	0
3	reserved	-	reserved	-

Table 934. Hash control register (CTRL) - Offset: 0h...continued

Bit	Name	Access	Description and value	Reset value
2:0	MODE	R/W	operational mode to use, or 0 if none Note: The CONFIG register indicate if specific modes beyond SHA1 and SHA2-256 are available.	0
			000b disabled	
			001b SHA1 is enabled	
			010b SHA2-256 is enabled	
			011b SHA2-512 is enabled	
			100b AES if available (see also Section 45.5.12.2.11 for more controls)	
			101b ICB-AES if available (see also Section 45.5.12.2.11 for more controls)	
			110b Salsa20/20 if available (including XSalsa – see also Section 45.5.12.2.11)	
			111b ChaCha20 if available (see also Section 45.5.12.2.11 for more controls)	

45.5.12.2.2 Status of hash peripheral (STATUS)

Table 935. Status of hash peripheral - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:22	reserved	-	reserved	-
21:16	ICBIDX	R	If ICB-AES is selected, reads as the ICB index count based on ICBSTRM (from CRYPTCFG) That is, if 3 bits of ICBSTRM, then this will count from 0 to 7 and then back to 0. On 0, it has to compute the full ICB, quicker when not 0.	0
15:10	reserved	-	reserved	-
9	PRNGFAULT	R	PRNG fault status	0
			0b no PRNG fault has occurred	
			1b PRNG fault has occurred	
8	AESFAULT	R	AES fault status	1
			0b no AES fault has occurred	
			1b AES fault has occurred	
7:6	reserved	-	reserved	-
5	NEEDIV	R	indicates the block wants an IV/NONE to be written in (set along with WAITING)	0
			0b no IV/nonce is necessary, either because written already or because it is not a requirement	
			1b IV/nonce is necessary and INDATA/ALIAS is accepted as IV/nonce Also sets WAITING.	

Table 935. Status of hash peripheral - Offset: 4h...continued

Bit	Name	Access	Description and value	Reset value
4	NEEDKEY	R	indicates the block wants the key to be written in (set along with WAITING)	0
			0b no key is required. Writes are not treated as key	
			1b Key is required. INDATA/ALIAS is accepted as Key Also sets WAITING.	
3	FAULT	R	indicates if an AES or PRNG fault has occurred	0
			0b no AES or PRNG fault has occurred	
			1b AES or PRNG fault has occurred	
2	ERROR	R	if 1, an error occurred For normal use, it is because of an attempted overrun. INDATA was written when it was not appropriate. For master cases, it is an AHB bus error. The COUNT field indicates which block it was on.	0
			0b no error	
			1b an error occurred since last cleared (written 1 to clear).	
1	DIGEST	R	For Hash, if 1, a DIGEST is ready and waiting and there is no active next block already started. For Cryptographic use, it is set for each block processed, indicating OUTDATA (and OUTDATA2 if larger output) contains the next value to read out. Cleared when any data is written, when new is written for cyptographic uses when the last word is read out, or when the block is disabled.	0
			0b no digest is ready	
			1b digest is ready Application may read it or may write more data.	
0	WAITING	R	If 1, the block is waiting for more data to process	0
			0b not waiting for data; may be disabled or busy Note: for cryptographic use, it is not set if <i>IsLast</i> is set nor is it going to be set until at least 1 word is read of the output.	
			1b waiting for data to be written (16 words)	

45.5.12.2.3 Interrupt enable set register (INTENSET)

Table 936. Interrupt enable set register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved	-
3	FAULT	R/W	Indicates if an interrupt must occur on an AES or PRNG fault as defined in Section 45.5.12.2.2 .	0
			0b no interrupt on an AES or PRNG fault	
			1b interrupt on an AES or PRNG fault	
2	ERROR	R/W	Indicates if an interrupt must occur on an ERROR as defined in Section 45.5.12.2.2 .	0
			0b no interrupt on error	
			1b interrupt on error	
1	DIGEST	R/W	Indicates if an interrupt must occur when digest (or outdata) is ready (completed a hash/crypto or completed a full sequence).	0
			0b no interrupt when digest is ready	
			1b interrupt when digest is ready Interrupt is cleared by writing more data, starting a new hash, or disabling (done).	
0	WAITING	R/W	Indicates if an interrupt must occur when waiting for data input.	0
			0b no interrupt when waiting	
			1b interrupt when waiting	

45.5.12.2.4 Interrupt enable clear register (INTENCLR)

Table 937. Interrupt enable clear register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:4	reserved	-	reserved	-
3	FAULT	R/W	write 1 to clear mask	0
2	ERROR	R/W	write 1 to clear mask	0
1	DIGEST	R/W	write 1 to clear mask	0
0	WAITING	R/W	write 1 to clear mask	0

45.5.12.2.5 Memory access control (MEMCTRL)

Table 938. Memory access control - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:27	reserved	-	reserved	-
26:16	COUNT	R/W	number of 512-bit (128-bit if AES, except 1st block which may include key and IV) blocks to copy starting at MEMADDR This register decrements after each block is copied, ending in 0. For hash, the DIGEST interrupt occurs when it reaches 0. For AES, the DIGEST/OUTDATA interrupt occurs on every block. If a bus error occurs, it stops with this field set to the block that failed. 0: Done – nothing to process. 1 K to 2 K: Number of 512-bit (or 128-bit) blocks to hash.	0
15:12	reserved	-	reserved	-
0	MASTER	R/W	enables mastering	0
			0b Mastering is not used and the normal DMA or interrupt based model is used with INDATA.	
			1b Mastering is enabled and DMA and INDATA must not be used.	

45.5.12.2.6 Start address for memory access (MEMADDR)

Table 939. Start address for memory access - Offset: 14h

Bit	Name	Access	Description and value	Reset value
31:0	BASE	R/W	address base to start copying from, word aligned (so bits 1:0 must be 0) This field advances as it processes the words. If it fails with a bus error, the register contains the failing word. N: Address in flash or RAM space; RAM only as mapped in this part. May also be able to address SPIFI.	0

45.5.12.2.7 16 words input data (INDATA)

Table 940. 16 words input data - Offset: 20h

Bit	Name	Access	Description and value	Reset value
31:0	DATA	W	write next word in little-endian form The hash requires big endian word data, but this block swaps the bytes automatically. That is, SHA assumes the data coming in is treated as bytes (for example, "abcd") and since the Arm core treats "abcd" as a word as 0x64636261, the block swaps the word to restore it into big endian.	0

45.5.12.2.8 Aliases for input data (ALIAS0 - ALIAS6)

Table 941. Offset

Register	Offset
ALIAS0	24h
ALIAS1	28h
ALIAS2	2Ch
ALIAS3	30h
ALIAS4	34h
ALIAS5	38h
ALIAS6	3Ch

Table 942. Aliases for input data

Bit	Name	Access	Description and value	Reset value
31:0	DATA	W	write next word in little-endian form The hash requires big endian word data, but this block swaps the bytes automatically. That is, SHA assumes the data coming in is treated as bytes (for example, "abcd") and since the Arm core treats "abcd" as a word as 0x64636261, the block swaps the word to restore it into big endian.	0

45.5.12.2.9 Result digest (DIGESTa)

For a = 0 to 7

Table 943. Result digest - Offset: 40h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	DIGEST	R	one word of the digest or output Note: Only the first 4 are populated for AES and the first 5 are populated for SHA1.	0

45.5.12.2.10 Result digest - part 2 (DIGESTa)

For a = 10 to 17

Table 944. Result digest - Offset: 60h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	DIGEST	R	One word of the second half of the output when used	0

45.5.12.2.11 Crypto configuration (CRYPTCFG)

Table 945. Crypto configuration - Offset: 80h

Bit	Name	Access	Description and value	Reset value
31:24	reserved	-	reserved	-
23:22	ICBSTRM	R/W	Size of the ICB-AES stream that can be pushed before needing to compute a new IV/ctr (counter start). Optimizes the performance of the stream of blocks after the first.	0
			00b 8 blocks	
			01b 16 blocks	
			10b 32 blocks	
			11b 64 blocks	
21:20	ICBSZ	R/W	Sets the ICB size between 32 and 128 bits, using the following rules. Note: The counter is assumed to occupy the low order bits of the IV.	0
			00b 32 bits of the IV/ctr are used (from 127:96)	
			01b 64 bits of the IV/ctr are used (from 127:64)	
			10b 96 bits of the IV/ctr are used (from 127:32)	
			11b All 128 bits of the IV/ctr are used	
19:18	reserved	-	reserved	-
17	XSALSA	R/W	Is 1 if XSalsa 128b NONCE must be used versus 64b.	0
16	STREAMLAST	R/W	Is 1 if last stream block. If not 1, then the engine computes the next "hash".	0
15:13	reserved	-	reserved	-
12:10	AESCTRPOS	R/W	halfword position of 16b counter in IV if AESMODE is CTR (position is fixed for Salsa and ChaCha) Only supports 16b counter, so the 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 must use ECB directly and do its own XOR, and so on.	0
9:8	AESKEYSZ	R/W	sets the AES key size	0
			00b 128 bit key	
			01b 192 bit key	
			10b 256 bit key	
			11b reserved	
7	AESSECRET	R/W	selects the hidden secret key versus user key, if provided If security levels are used, only the highest level is permitted to select this option.	0
			0b user key provided in normal way	
			1b secret key provided in hidden way by HW	

Table 945. Crypto configuration - Offset: 80h...continued

Bit	Name	Access	Description and value	Reset value
6	AESDECRYPT	R/W	AES ECB direction Encryption is only used if in CTR mode or manual modes, such as CFB.	0
			0b encrypt	
			1b decrypt	
5:4	AESMODE	R/W	AES cipher mode to use if plain AES	0
			00b ECB – used as is	
			01b CBC mode (see details on IV/nonce)	
			10b CTR mode (see details on IV/nonce; see also AESCTRPOS)	
			11b reserved	
3	MSW1ST	R/W	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.	0
2	SWAPDAT	R/W	if 1, swaps the data and IV inputs (bytes in each word)	0
1	SWAPKEY	R/W	if 1, swaps the key input (bytes in each word)	0
0	MSW1ST_OUT	R/W	If 1, OUTDATA0 is read: Most significant word first for AES. Else, it is read in normal little endian - least significant word first. Note: Only if allowed by configuration.	

45.5.12.2.12 Hardware configuration (CONFIG)

Table 946. Hardware configuration - Offset: 84h

Bit	Name	Access	Description and value	Reset value
31:12	reserved	-	reserved	-
11	ICB	R	1 if ICB over AES included	0
10	CHACHA	R	1 if CHACHA is included	0
9	SALSA	R	1 if SALSA is included	0
8	SECRET	R	1 if secret key is available	0
7	AESKEY	R	1 if AES 192 and 256 are also included	0
6	AES	R	1 if AES 128 is included	0
5	SHA512	R	1 if SHA2-512 is included	0
4	reserved	-	reserved	-
3	AHB	R	1 if AHB master is enabled	0
2	reserved	-	reserved	-
1	DMA	R	1 if DMA is connected	0
0	DUAL	R	1 if 2 x 512 bit buffers are included; 0 if only 1 x 512 bit is included	0

45.5.12.2.13 Register lock (LOCK)

Table 947. Register lock - Offset: 8Ch

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved	-
15:4	PATTERN	R/W	must write 0xA75 to change lock state A75: Pattern required to change bits 1:0	0
3:2	reserved	-	reserved	-
1:0	SECLOCK	R/W	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 or higher security level as lock. Reads as: 0 if unlocked, else 1, 2, 3 to indicate security level it is locked at. Note: This register and the ID register are the only readable registers if locked and current state is lower than lock level.	0

45.5.12.2.14 ICB mask (MASK0 - MASK3)

Table 948. Offset

Register	Offset
MASK0	90h
MASK1	94h
MASK2	98h
MASK3	9Ch

Table 949. ICB mask

Bit	Name	Access	Description and value	Reset value
31:0	MASK	W	random word	0

45.5.12.2.15 SHA digest to reload (RELOADa)

For a = 0 to 7

Table 950. SHA digest to reload - Offset: A0h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	DIGEST	W	SHA digest word to reload	0

45.5.12.2.16 Pseudo random number generator seed value (PRNG_SEED)

This input offers random seed for the PRNG block that is used for masking in AES operations. This register is written at the start of an AES operation and must 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.

Table 951. Pseudo random number generator seed value - Offset: D0h

Bit	Name	Access	Description and value	Reset value
31:0	PRNG_SEED	W	random input value used as an entropy source	0

45.5.12.2.17 Pseudo random number (PRNG_OUT)

This register provides a 32-bit pseudo random number. This register is not accessible when AES engine is in use, it is set to 0x00000000.

Table 952. Pseudo random number - Offset: D8h

Bit	Name	Access	Description and value	Reset value
31:0	PRNG_OUT	R	provides random number	0

45.5.13 Crypto-coprocessor (CASPER)

45.5.13.1 CASPER memory map

The CASPER registers are located at base address: 400A_5000h.

Table 953. Crypto-coprocessor registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CTRL0	R/W	0h	offsets of AB and CD in RAM	32		Section 45.5.13.2.1
CTRL1	R/W	4h	opcode mode, iteration count, and result offset (in RAM); also launches the accelerator	32		Section 45.5.13.2.2
LOADER	R/W	8h	optional loader to load into CTRL0/1 in steps; to perform a set of operations	32		Section 45.5.13.2.3
STATUS	R/W	Ch	indicates operational status; contains the carry bit if used	32		Section 45.5.13.2.4
INTENSET	R/W	10h	interrupt enable set register	32		Section 45.5.13.2.5
INTENCLR	W	14h	interrupt enable clear register	32		Section 45.5.13.2.6
INTSTAT	W	18h	interrupt status bits	32		Section 45.5.13.2.7
AREG	R/W	20h	a register	32	0000_0000h	Section 45.5.13.2.8
BREG	R/W	24h	b register	32	0000_0000h	Section 45.5.13.2.9
CREG	R/W	28h	c register	32	0000_0000h	Section 45.5.13.2.10
DREG	R/W	2Ch	d register	32	0000_0000h	Section 45.5.13.2.11
RES0	R/W	30h	result register 0	32	0000_0000h	Section 45.5.13.2.12
RES1	R/W	34h	result register 1	32	0000_0000h	Section 45.5.13.2.13
RES2	R/W	38h	result register 2	32	0000_0000h	Section 45.5.13.2.14
RES3	R/W	3Ch	result register 3	32	0000_0000h	Section 45.5.13.2.15
MASK	R/W	60h	optional mask register	32	0000_0000h	Section 45.5.13.2.16
REMASK	R/W	64h	optional remask register	32	0000_0000h	Section 45.5.13.2.17
LOCK	R/W	80h	security lock register	32		Section 45.5.13.2.18

45.5.13.2 Register descriptions

45.5.13.2.1 Control 0 register (CTRL0)

Table 954. Offsets of AB and CD in the RAM - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:29	reserved	-	reserved	-
28:18	CDOFF	R/W	Word or DWord offset of CD, with D at [2] = 0 and C at [2] = 1 as far as the code sees (normally is an interleaved bank so only sequential to AHB) Word offset is only allowed in 32-bit operation. Ideally, not in the same RAM as the AB values.	0
17	reserved	-	reserved	-
16	CDBPAIR	R/W	bank-pair the offset CDOFF is in 0b bank-pair 0 (1st) 1b bank-pair 1 (2nd)	0
15:13	reserved	-	reserved	-
12:2	ABOFF	R/W	Word or DWord offset of AB values, with B at [2] = 0 and A at [2] = 1 as far as the code sees (normally is an interleaved bank so only sequential to AHB) Word offset only allowed in 32-bit operation. Ideally, not in the same RAM as the CD values if 4-up.	0
1	reserved	-	reserved	-
0	ABBPAIR	R/W	bank-pair the offset ABOFF is in Must be 0 if only 2-up. 0b bank-pair 0 (1st) 1b bank-pair 1 (2nd)	0

45.5.13.2.2 Control 1 register (CTRL1)

Table 955. Control 1 register - Offset: 4h

Bit	Name	Access	Description and value	Reset value
31:30	CSKIP	R/W	skip rules on carry if necessary This operation is skipped based on carry value (from previous operation) if not 0: 00b no skip 01b skip if carry is 1 10b skip if carry is 0 11b set CTRL0FF to CDOFF and skip	0
29	reserved	-	reserved	-
28:18	RESOFF	R/W	Word or DWord offset of result Word offset only allowed in 32-bit operation. Ideally, not in the same RAM as the AB and CD values	0

Table 955. Control 1 register - Offset: 4h...continued

Bit	Name	Access	Description and value	Reset value
17	reserved	-	reserved	-
16	RESBPAIR	R/W	bank-pair the offset RESOFF is in Must be 0 if only 2-up. Ideally, not the samebank as ABBPAIR (when 4-up supported).	0
			0b bank-pair 0 (1st)	
			1b bank-pair 1 (2nd)	
15:8	MODE	R/W	operation mode to perform Writing 0 means that the accelerator is inactive. Writing others means that the accelerator is active.	0
0	ITER	R/W	iteration counter Is number_cycles - 1. writing 0 means: Does one cycle - does not iterate.	0

45.5.13.2.3 Loader register (LOADER)

Table 956. Loader register - Offset: 8h

Bit	Name	Access	Description and value	Reset value
31:29	reserved	-	reserved	-
28:18	CTRLOFF	R/W	DWord offset of CTRL pair to load next	0
17	reserved	-	reserved	-
16	CTRLBPAIR	R/W	bank-pair the offset CTRLOFF is in Must be 0 if only 2-up. Does not matter which bank is used as it is loaded when not performing an operation.	0
			0b bank-pair 0 (1st)	
			1b bank-pair 1 (2nd)	
15:8	reserved	-	reserved	-
7:0	COUNT	R/W	number of control pairs to load 0 relative (so 1 means load 1) Writing 1 means: Does one op - does not iterate. Writing N means N control pairs to load.	0

45.5.13.2.4 Status register (STATUS)

Table 957. Status register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved	-
5	BUSY	R	Indicates if the accelerator is busy performing an operation	0
			0b not busy - is idle	
			1b is busy	
4	CARRY	R	last carry value if operation produced a carry bit	0
			0b carry was 0 or no carry	
			1b carry was 1	
3:1	reserved	-	reserved	-
0	DONE	R/W	indicates if the accelerator has finished an operation Write 1 to clear or write CTRL1 to clear.	0

45.5.13.2.5 Interrupt enable set register (INTENSET)

Table 958. Interrupt enable set register - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved	-
0	DONE	R/W	set if the accelerator should interrupt when done	0
			0b do not interrupt when done	
			1b interrupt when done	

45.5.13.2.6 Interrupt enable clear register (INTENCLR)

Table 959. Interrupt enable clear register - Offset: 14h

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved	-
0	DONE	R/W	written to clear an interrupt set with INTENSET	0
			0b if written 0, ignored	
			1b if written 1, do not interrupt when done	

45.5.13.2.7 Interrupt status bits (INTSTAT)

Table 960. Interrupt status bits - Offset: 18h

Bit	Name	Access	Description and value	Reset value
31:1	reserved	-	reserved	-
0	DONE	R/W	If set, interrupt because by the accelerator being done	0
			0b not caused by the accelerator being done	
			1b caused by the accelerator being done	

45.5.13.2.8 A register (AREG)

Table 961. A register - Offset: 20h

Bit	Name	Access	Description and value	Reset value
31:0	REG_VALUE	R/W	register to be fed into multiplier Is not normally written or read by application, but is available when accelerator is not busy.	0

45.5.13.2.9 B register (BREG)

Table 962. B register - Offset: 24h

Bit	Name	Access	Description and value	Reset value
31:0	REG_VALUE	R/W	register to be fed into multiplier Is not normally written or read by application, but is available when accelerator is not busy.	0

45.5.13.2.10 C register (CREG)

Table 963. C register - Offset: 28h

Bit	Name	Access	Description and value	Reset value
31:0	REG_VALUE	R/W	register to be fed into multiplier Is not normally written or read by application, but is available when accelerator is not busy.	0

45.5.13.2.11 D register (DREG)

Table 964. D register - Offset: 28h

Bit	Name	Access	Description and value	Reset value
31:0	REG_VALUE	R/W	register to be fed into multiplier Is not normally written or read by application, but is available when accelerator is not busy.	0

45.5.13.2.12 Result register 0 (RES0)

Table 965. Result register 0 - Offset: 30h

Bit	Name	Access	Description and value	Reset value
31:0	REG_VALUE	R/W	register to hold working result (from multiplier, adder/xor, and so on) Is not normally written or read by application, but is available when accelerator is not busy.	0

45.5.13.2.13 Result register 1 (RES1)

Table 966. Result register 1 - Offset: 34h

Bit	Name	Access	Description and value	Reset value
31:0	REG_VALUE	R/W	register to hold working result (from multiplier, adder/xor, and so on) Is not normally written or read by application, but is available when accelerator is not busy.	0

45.5.13.2.14 Result register 2 (RES2)

Table 967. Result register 2 - Offset: 38h

Bit	Name	Access	Description and value	Reset value
31:0	REG_VALUE	R/W	register to hold working result (from multiplier, adder/xor, and so on) Is not normally written or read by application, but is available when accelerator is not busy.	0

45.5.13.2.15 Result register 3 (RES3)

Table 968. Result register 3 - Offset: 3Ch

Bit	Name	Access	Description and value	Reset value
31:0	REG_VALUE	R/W	register to hold working result (from multiplier, adder/xor, and so on) Is not normally written or read by application, but is available when accelerator is not busy.	0

45.5.13.2.16 Optional mask register (MASK)

Table 969. Optional mask register - Offset: 60h

Bit	Name	Access	Description and value	Reset value
31:0	MASK	R/W	mask to apply as side channel countermeasure	0
			0	no mask to be used
			N	mask to XOR onto values

45.5.13.2.17 Optional remask register (REMASK)

Table 970. Optional remask register - Offset: 64h

Bit	Name	Access	Description and value	Reset value
31:0	MASK	R/W	mask to apply as side channel countermeasure	0
			0 no mask to be used	
			N mask to XOR onto values	

45.5.13.2.18 Security lock register (LOCK)

Table 971. Security lock register - Offset: 80h

Bit	Name	Access	Description and value	Reset value
31:17	reserved	-	reserved	-
16:4	KEY	R/W	must be written as 0x73D to change the register 0_0111_0011_1101b: If set during write, allows lock or unlock.	0
3:1	reserved	-	reserved	-
0	LOCK	R/W	Reads back with security level locked to, or 0 Writes 0 to unlock; writes 1 to lock.	0
			0b unlock	
			1b lock to current security level	

45.5.14 AHB security control (AHB_SECURE_CTRL)

45.5.14.1 Security control memory map

The AHB_SECURE_CTRL registers are located at base address: 400A_C000h.

Table 972. AHB secure controller registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
SEC_CTRL_FLASH_ROM_SLAVE_RULE	R/W	0h	security access rules for flash and ROM slaves	32	0000_0000h	Section 45.5.14.2.1
SEC_CTRL_FLASH_MEM_RULE0 - SEC_CTRL_FLASH_MEM_RULE2	R/W	10h - 18h	security access rules for FLASH sector 0 to sector 16	32	0000_0000h	Section 45.5.14.2.2
SEC_CTRL_ROM_MEM_RULE0 - SEC_CTRL_ROM_MEM_RULE3	R/W	20h - 2Ch	security access rules for ROM sector 0 to sector 31	32	0000_0000h	Section 45.5.14.2.3
SEC_CTRL_RAMX_SLAVE_RULE	R/W	30h	security access rules for RAMX slaves	32	0000_0000h	Section 45.5.14.2.4
SEC_CTRL_RAMX_MEM_RULE0	R/W	40h	security access rules for RAMX memory	32	0000_0000h	Section 45.5.14.2.5
SEC_CTRL_RAM0_SLAVE_RULE	R/W	50h	security access rules for RAM0 slave	32	0000_0000h	Section 45.5.14.2.6
SEC_CTRL_RAM0_MEM_RULE0	R/W	60h	security access rules for RAM0 memory	32	0000_0000h	Section 45.5.14.2.7
SEC_CTRL_RAM1_SLAVE_RULE	R/W	70h	security access rules for RAM1 slave	32	0000_0000h	Section 45.5.14.2.8
SEC_CTRL_RAM1_MEM_RULE0	R/W	80h	security access rules for RAM1 memory	32	0000_0000h	Section 45.5.14.2.9
SEC_CTRL_RAM2_SLAVE_RULE	R/W	90h	security access rules for RAM2 slave	32	0000_0000h	Section 45.5.14.2.10
SEC_CTRL_RAM2_MEM_RULE0 - SEC_CTRL_RAM2_MEM_RULE1	R/W	A0h - A4h	security access rules for RAM2 memory	32	0000_0000h	Section 45.5.14.2.11
SEC_CTRL_SPIFI_SLAVE_RULE	R/W	B0h	security access rules for SPIFI slave	32	0000_0000h	Section 45.5.14.2.12
SEC_CTRL_SPIFI_MEM_RULE0 - SEC_CTRL_SPIFI_MEM_RULE3	R/W	C0h - CCh	security access rules for SPIFI memory	32	0000_0000h	Section 45.5.14.2.13
SEC_CTRL_APB_BRIDGE_SLAVE_RULE	R/W	D0h	security access rules for both APB bridge slaves	32	0000_0000h	Section 45.5.14.2.14
SEC_CTRL_APB_BRIDGE0_MEM_CTRL0	R/W	E0h	security access rules for APB bridge 0 peripherals	32	0000_0000h	Section 45.5.14.2.15
SEC_CTRL_APB_BRIDGE0_MEM_CTRL1	R/W	E4h	security access rules for APB bridge 0 peripherals	32	0000_0000h	Section 45.5.14.2.16
SEC_CTRL_APB_BRIDGE0_MEM_CTRL2	R/W	E8h	security access rules for APB bridge 0 peripherals	32	0000_0000h	Section 45.5.14.2.17

Table 972. AHB secure controller registers...continued

Name	Access	Offset	Description	Width (bits)	Reset value	Section
SEC_CTRL_APB_BRIDGE0_MEM_CTRL3	R/W	ECh	security access rules for APB bridge 0 peripherals	32	0000_0000h	Section 45.5.14.2.18
SEC_CTRL_APB_BRIDGE1_MEM_CTRL0	R/W	F0h	security access rules for APB bridge 1 peripherals	32	0000_0000h	Section 45.5.14.2.19
SEC_CTRL_APB_BRIDGE1_MEM_CTRL1	R/W	F4h	security access rules for APB bridge 1 peripherals	32	0000_0000h	Section 45.5.14.2.20
SEC_CTRL_APB_BRIDGE1_MEM_CTRL2	R/W	F8h	security access rules for APB bridge 1 peripherals	32	0000_0000h	Section 45.5.14.2.21
SEC_CTRL_APB_BRIDGE1_MEM_CTRL3	R/W	FC h	security access rules for APB bridge 1 peripherals	32	0000_0000h	Section 45.5.14.2.22
SEC_CTRL_AHB_PORT7_SLAVE0_RULE	R/W	100h	security access rules for AHB peripheral on port 7 - slave 0	32	0000_0000h	Section 45.5.14.2.23
SEC_CTRL_AHB_PORT7_SLAVE1_RULE	R/W	104h	security access rules for AHB peripheral on port 7 - slave 1	32	0000_0000h	Section 45.5.14.2.24
SEC_CTRL_AHB_PORT8_SLAVE0_RULE	R/W	110h	security access rules for AHB peripherals on port 8 - slave 0	32	0000_0000h	Section 45.5.14.2.25
SEC_CTRL_AHB_PORT8_SLAVE1_RULE	R/W	114h	security access rules for AHB peripherals on port 8 - slave 1	32	0000_0000h	Section 45.5.14.2.26
SEC_CTRL_AHB_PORT9_SLAVE0_RULE	R/W	120h	security access rules for AHB peripherals on port 9 - slave 0	32	0000_0000h	Section 45.5.14.2.27
SEC_CTRL_AHB_PORT9_SLAVE1_RULE	R/W	124H	security access rules for AHB peripherals on port 9 - slave 1	32	0000_0000H	Section 45.5.14.2.28
SEC_CTRL_AHB_SEC_CTRL_MEM_RULE	R/W	130h	security access rules for AHB_SEC_CTRL memory	32	0000_0000h	Section 45.5.14.2.29
SEC_CTRL_AHB_PORT10_SLAVE0_RULE	R/W	140h	security access rules for AHB peripherals on port 10 - slave 0	32	0000_0000h	Section 45.5.14.2.30
SEC_VIO_ADDR0 - SEC_VIO_ADDR9	R	E00h - E24h	most recent security violation address for AHB layer n	32	0000_0000h	Section 45.5.14.2.31
SEC_VIO_MISC_INFO0 - SEC_VIO_MISC_INFO9	R	E80h - EA4h	most recent security violation miscellaneous information for AHB	32		Section 45.5.14.2.32
SEC_VIO_INFO_VALID	R/W	F00h	security violation address/information registers valid flags	32		Section 45.5.14.2.33
SEC_GPIO_MASK0	R/W	F80h	secure GPIO mask for port 0 pins	32	007F_FFFFh	Section 45.5.14.2.34

Table 972. AHB secure controller registers...continued

Name	Access	Offset	Description	Width (bits)	Reset value	Section
SEC_MASK_LOCK	R/W	FBCh	security general-purpose register access control	32		Section 45.5.14.2.35
MASTER_SEC_LEVEL	R/W	FD0h	master secure level register	32	8000_0000h	Section 45.5.14.2.36
MASTER_SEC_ANTI_POL_REG	R/W	FD4h	master secure level anti-pole register	32	BFFF_FFFFh	Section 45.5.14.2.37
CPU0_LOCK_REG	R/W	FECh	miscellaneous control signals for in Cortex M33 (CPU0)	32		Section 45.5.14.2.38
MISC_CTRL_DP_REG	R/W	FF8h	secure control duplicate register	32		Section 45.5.14.2.39
MISC_CTRL_REG	R/W	FFCh	secure control register	32		Section 45.5.14.2.40

45.5.14.2 Register descriptions

45.5.14.2.1 Security access rules for flash and ROM slaves (SEC_CTRL_FLASH_ROM_SLAVE_RULE)

Table 973. Security access rules for flash and ROM slaves - Offset: 0h

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved	-
5:4	ROM_RULE	R/W	security access rules for the whole ROM: 0x0300_0000 - 0x0301_FF FF	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
3:2	reserved	-	reserved	-
1:0	FLASH_RULE	R/W	security access rules for the whole FLASH: 0x0000_0000 - 0x0010_9 FFF	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.2 Security access rules for FLASH sector 0 to sector 16 (SEC_CTRL_FLASH_MEM_RULE0 - SEC_CTRL_FLASH_MEM_RULE2)

Each flash sector is 64 Kbytes. There are 17 FLASH sectors in total:

- mem rule 0_0 address space: 0x0000_0000 - 0x0000_FFFF
- mem rule 0_1 address space: 0x0001_0000 - 0x0001_FFFF
- mem rule 0_2 address space: 0x0002_0000 - 0x0002_FFFF
- mem rule 0_3 address space: 0x0003_0000 - 0x0003_FFFF
- mem rule 0_4 address space: 0x0004_0000 - 0x0004_FFFF
- mem rule 0_5 address space: 0x0005_0000 - 0x0005_FFFF
- mem rule 0_6 address space: 0x0006_0000 - 0x0006_FFFF
- mem rule 0_7 address space: 0x0007_0000 - 0x0007_FFFF
- mem rule 1_0 address space: 0x0008_0000 - 0x0008_FFFF
- mem rule 1_1 address space: 0x0009_0000 - 0x0009_FFFF
- mem rule 1_2 address space: 0x000A_0000 - 0x000A_FFFF
- mem rule 1_3 address space: 0x000B_0000 - 0x000B_FFFF
- mem rule 1_4 address space: 0x000C_0000 - 0x000C_FFFF
- mem rule 1_5 address space: 0x000D_0000 - 0x000D_FFFF
- mem rule 1_6 address space: 0x000E_0000 - 0x000E_FFFF
- mem rule 1_7 address space: 0x000F_0000 - 0x000F_FFFF
- mem rule 2_0 address space: 0x0010_0000 - 0x0010_FFFF

Table 974. Offset

Register	Offset
SEC_CTRL_FLASH_MEM_RULE0	10h
SEC_CTRL_FLASH_MEM_RULE1	14h
SEC_CTRL_FLASH_MEM_RULE2	18h

Table 975. Security access rules for FLASH sector 0 to sector 16

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved	-
29:28	RULE7	R/W	secure control rule7 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
27:26	reserved	-	reserved	-

Table 975. Security access rules for FLASH sector 0 to sector 16...continued

Bit	Name	Access	Description and value	Reset value
25:24	RULE6	R/W	secure control rule6 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
23:22	reserved	-	reserved	-
21:20	RULE5	R/W	secure control rule5 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
19:18	reserved	-	reserved	-
17:16	RULE4	R/W	secure control rule4 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
15:14	reserved	-	reserved	-
13:12	RULE3	R/W	secure control rule3 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
11:10	reserved	-	reserved	-
9:8	RULE2	R/W	secure control rule2 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
7:6	reserved	-	reserved	-

Table 975. Security access rules for FLASH sector 0 to sector 16...continued

Bit	Name	Access	Description and value	Reset value
5:4	RULE1	R/W	secure control rule1 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
3:2	reserved	-	reserved	-
1:0	RULE0	R/W	secure control rule0 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.3 Security access rules for ROM sector 0 to sector 31 (SEC_CTRL_ROM_MEM_RULE0 - SEC_CTRL_ROM_MEM_RULE3)

Each ROM sector is 8 Kbytes. There are 32 ROM sectors in total:

- mem rule 0_0 address space: 0x0300_0000 - 0x0300_1FFF
- mem rule 0_1 address space: 0x0300_2000 - 0x0300_3FFF
- mem rule 0_2 address space: 0x0300_4000 - 0x0300_5FFF
- mem rule 0_3 address space: 0x0300_6000 - 0x0300_7FFF
- mem rule 0_4 address space: 0x0300_8000 - 0x0300_9FFF
- mem rule 0_5 address space: 0x0300_A000 - 0x0300_BFFF
- mem rule 0_6 address space: 0x0300_C000 - 0x0300_DFFF
- mem rule 0_7 address space: 0x0300_E000 - 0x0300_FFFF
- mem rule 1_0 address space: 0x0301_0000 - 0x0301_1FFF
- mem rule 1_1 address space: 0x0301_2000 - 0x0301_3FFF
- mem rule 1_2 address space: 0x0301_4000 - 0x0301_5FFF
- mem rule 1_3 address space: 0x0301_6000 - 0x0301_7FFF
- mem rule 1_4 address space: 0x0301_8000 - 0x0301_9FFF
- mem rule 1_5 address space: 0x0301_A000 - 0x0301_BFFF
- mem rule 1_6 address space: 0x0301_C000 - 0x0301_DFFF
- mem rule 1_7 address space: 0x0301_E000 - 0x0301_FFFF
- mem rule 2_0 address space: 0x0302_0000 - 0x0302_1FFF
- mem rule 2_1 address space: 0x0302_2000 - 0x0302_3FFF
- mem rule 2_2 address space: 0x0302_4000 - 0x0302_5FFF
- mem rule 2_3 address space: 0x0302_6000 - 0x0302_7FFF
- mem rule 2_4 address space: 0x0302_8000 - 0x0302_9FFF
- mem rule 2_5 address space: 0x0302_A000 - 0x0302_BFFF
- mem rule 2_6 address space: 0x0302_C000 - 0x0302_DFFF
- mem rule 2_7 address space: 0x0302_E000 - 0x0302_FFFF
- mem rule 3_0 address space: 0x0303_0000 - 0x0303_1FFF
- mem rule 3_1 address space: 0x0303_2000 - 0x0303_3FFF
- mem rule 3_2 address space: 0x0303_4000 - 0x0303_5FFF
- mem rule 3_3 address space: 0x0303_6000 - 0x0303_7FFF
- mem rule 3_4 address space: 0x0303_8000 - 0x0303_9FFF
- mem rule 3_5 address space: 0x0303_A000 - 0x0303_BFFF
- mem rule 3_6 address space: 0x0303_C000 - 0x0303_DFFF
- mem rule 3_7 address space: 0x0303_E000 - 0x0303_FFFF

Table 976. Offset

Register	Offset
SEC_CTRL_ROM_MEM_RULE0	20h
SEC_CTRL_ROM_MEM_RULE1	24h
SEC_CTRL_ROM_MEM_RULE2	28h
SEC_CTRL_ROM_MEM_RULE3	2Ch

Table 977. Security access rules for ROM sector 0 to sector 31

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved	-
29:28	RULE7	R/W	secure control rule7 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
27:26	reserved	-	reserved	-
25:24	RULE6	R/W	secure control rule6 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
23:22	reserved	-	reserved	-
21:20	RULE5	R/W	secure control rule5 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
19:18	reserved	-	reserved	-
17:16	RULE4	R/W	secure control rule4 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
15:14	reserved	-	reserved	-
13:12	RULE3	R/W	secure control rule3 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
11:10	reserved	-	reserved	-

Table 977. Security access rules for ROM sector 0 to sector 31...continued

Bit	Name	Access	Description and value	Reset value
9:8	RULE2	R/W	secure control rule2 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
7:6	reserved	-	reserved	-
5:4	RULE1	R/W	secure control rule1 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
3:2	reserved	-	reserved	-
1:0	RULE0	R/W	secure control rule0 can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.4 Security access rules for RAMX slave (SEC_CTRL_RAMX_SLAVE_RULE)

Table 978. Security access rules for RAMX slave - Offset: 30h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1:0	RAMX_RULE	R/W	security access rules for the whole RAMX: 0x0400_0000 - 0x0400_7FFF	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.5 Security access rules for RAMX memory (SEC_CTRL_RAMX_MEM_RULE0)

Security access rules for RAMX subregion. Each RAMX subregion is 4 kbytes.

- 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

Table 979. Security access rules for RAMX memory - Offset: 40h

Bit	Name	Access	Description and value	Reset value
31:14	reserved	-	reserved	-
13:12	RULE3	R/W	secure control rule3 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
11:10	reserved	-	reserved	-
9:8	RULE2	R/W	secure control rule2 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
7:6	reserved	-	reserved	-
5:4	RULE1	R/W	secure control rule1 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
3:2	reserved	-	reserved	-
1:0	RULE0	R/W	secure control rule0 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.6 Security access rules for RAM0 slave (SEC_CTRL_RAM0_SLAVE_RULE)

Table 980. Security access rules for RAM0 slave - Offset: 50h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1:0	RAM0_RULE	R/W	security access rules for the whole RAM0: 0x2000_4000 - 0x2000_7FFF	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.7 Security access rules for RAM0 memory (SEC_CTRL_RAM0_MEM_RULE0)

Security access rules for RAM0 subregion. Each RAM0 subregion is 4 kbytes.

- mem rule 0_0 Address space: 0x2000_4000 - 0x2000_4FFF
- mem rule 0_1 Address space: 0x2000_5000 - 0x2000_5FFF
- mem rule 0_2 Address space: 0x2000_6000 - 0x2000_6FFF
- mem rule 0_3 Address space: 0x2000_7000 - 0x2000_7FFF

Table 981. Security access rules for RAM0 memory - Offset: 60h

Bit	Name	Access	Description and value	Reset value
31:14	reserved	-	reserved	-
13:12	RULE3	R/W	secure control rule3 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
11:10	reserved	-	reserved	-
9:8	RULE2	R/W	secure control rule2 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
7:6	reserved	-	reserved	-

Table 981. Security access rules for RAM0 memory - Offset: 60h...continued

Bit	Name	Access	Description and value	Reset value
5:4	RULE1	R/W	secure control rule1 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
3:2	reserved	-	reserved	-
1:0	RULE0	R/W	secure control rule0 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.8 Security access rules for RAM1 slave (SEC_CTRL_RAM1_SLAVE_RULE)

Table 982. Security access rules for RAM1 slave - Offset: 70h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1:0	RAM1_RULE	R/W	security access rules for the whole RAM1: 0x2000_8000 - 0x2000_FF FF	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.9 Security access rules for RAM1 memory (SEC_CTRL_RAM1_MEM_RULE0)

Security access rules for RAM1 subregion. Each RAM1 subregion is 4 kbytes.

- 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
- mem rule 0_4 Address space: 0x2000_C000 - 0x2000_CFFF
- mem rule 0_5 Address space: 0x2000_D000 - 0x2000_DFFF
- mem rule 0_6 Address space: 0x2000_E000 - 0x2000_EFFF
- mem rule 0_7 Address space: 0x2000_F000 - 0x2000_FFFF

Table 983. Security access rules for RAM0 memory - Offset: 60h

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved	-
29:28	RULE7	R/W	secure control rule7 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
27:26	reserved	-	reserved	-
25:24	RULE6	R/W	secure control rule6 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
23:22	reserved	-	reserved	-
21:20	RULE5	R/W	secure control rule5 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
19:18	reserved	-	reserved	-

Table 983. Security access rules for RAM0 memory - Offset: 60h...continued

Bit	Name	Access	Description and value	Reset value
17:16	RULE4	R/W	secure control rule4 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
15:14	reserved	-	reserved	-
13:12	RULE3	R/W	secure control rule3 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
11:10	reserved	-	reserved	-
9:8	RULE2	R/W	secure control rule2 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
7:6	reserved	-	reserved	-
5:4	RULE1	R/W	secure control rule1 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
3:2	reserved	-	reserved	-
1:0	RULE0	R/W	secure control rule0 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.10 Security access rules for RAM2 slave (SEC_CTRL_RAM2_SLAVE_RULE)

Table 984. Security access rules for RAM2 slave - Offset: 90h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1:0	RAM2_RULE	R/W	Security access rules for the whole RAM2: 0x2001_0000 - 0x2001_FF FF	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.11 Security access rules for RAM2 memory (SEC_CTRL_RAM2_MEM_RULE0 - SEC_CTRL_RAM2_MEM_RULE1)

Security access rules for RAM2 subregion. Each RAM2 subregion is 4 kbytes.

- mem rule 0_0 Address space: 0x2001_0000 - 0x2001_0FFF
- mem rule 0_1 Address space: 0x2001_1000 - 0x2001_1FFF
- mem rule 0_2 Address space: 0x2001_2000 - 0x2001_2FFF
- mem rule 0_3 Address space: 0x2001_3000 - 0x2001_3FFF
- mem rule 0_4 Address space: 0x2001_4000 - 0x2001_4FFF
- mem rule 0_5 Address space: 0x2001_5000 - 0x2001_5FFF
- mem rule 0_6 Address space: 0x2001_6000 - 0x2001_6FFF
- mem rule 0_7 Address space: 0x2001_7000 - 0x2001_7FFF
- mem rule 1_0 Address space: 0x2001_8000 - 0x2001_8FFF
- mem rule 1_1 Address space: 0x2001_9000 - 0x2001_9FFF
- mem rule 1_2 Address space: 0x2001_A000 - 0x2001_AFFF
- mem rule 1_3 Address space: 0x2001_B000 - 0x2001_BFFF
- mem rule 1_4 Address space: 0x2001_C000 - 0x2001_CFFF
- mem rule 1_5 Address space: 0x2001_D000 - 0x2001_DFFF
- mem rule 1_6 Address space: 0x2001_E000 - 0x2001_EFFF
- mem rule 1_7 Address space: 0x2001_F000 - 0x2001_FFFF

Table 985. Offset

Register	Offset
SEC_CTRL_RAM2_MEM_RULE0	A0h
SEC_CTRL_RAM2_MEM_RULE1	A4h

Table 986. Security access rules for RAM0 memory

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved	-
29:28	RULE7	R/W	secure control rule7 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
27:26	reserved	-	reserved	-
25:24	RULE6	R/W	secure control rule6 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
23:22	reserved	-	reserved	-
21:20	RULE5	R/W	secure control rule5 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
19:18	reserved	-	reserved	-
17:16	RULE4	R/W	secure control rule4 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
15:14	reserved	-	reserved	-
13:12	RULE3	R/W	secure control rule3 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
11:10	reserved	-	reserved	-

Table 986. Security access rules for RAM0 memory...continued

Bit	Name	Access	Description and value	Reset value
9:8	RULE2	R/W	secure control rule2 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
7:6	reserved	-	reserved	-
5:4	RULE1	R/W	secure control rule1 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
3:2	reserved	-	reserved	-
1:0	RULE0	R/W	secure control rule0 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.12 Security access rules for SPIFI slave (SEC_CTRL_SPIFI_SLAVE_RULE)

Table 987. Security access rules for SPIFI slave - Offset: B0h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1:0	SPIFI_RULE	R/W	Security access rules for the whole SPIFI: 0x2800_0000 - 0x28FF_FFF	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.13 Security access rules for SPIFI memory (SEC_CTRL_SPIFI_MEM_RULE0 - SEC_CTRL_SPIFI_MEM_RULE3)

Security access rules for SPI flash interface subregion. There are 32 subregions. Each subregion is 4 Mbytes.

- mem rule 0_0 Address space: 0x2800_0000 - 0x283F_FFFF
- mem rule 0_1 Address space: 0x2840_0000 - 0x287F_FFFF
- mem rule 0_2 Address space: 0x2880_0000 - 0x28BF_FFFF
- mem rule 0_3 Address space: 0x28C0_0000 - 0x28FF_FFFF
- mem rule 0_4 Address space: 0x2900_0000 - 0x293F_FFFF
- mem rule 0_5 Address space: 0x2940_0000 - 0x297F_FFFF
- mem rule 0_6 Address space: 0x2980_0000 - 0x29BF_FFFF
- mem rule 0_7 Address space: 0x29C0_0000 - 0x29FF_FFFF
- mem rule 1_0 Address space: 0x2A00_0000 - 0x2A3F_FFFF
- mem rule 1_1 Address space: 0x2A40_0000 - 0x2A7F_FFFF
- mem rule 1_2 Address space: 0x2A80_0000 - 0x2ABF_FFFF
- mem rule 1_3 Address space: 0x2AC0_0000 - 0x2AFF_FFFF
- mem rule 1_4 Address space: 0x2B00_0000 - 0x2B3F_FFFF
- mem rule 1_5 Address space: 0x2B40_0000 - 0x2B7F_FFFF
- mem rule 1_6 Address space: 0x2B80_0000 - 0x2BBF_FFFF
- mem rule 1_7 Address space: 0x2BC0_0000 - 0x2BFF_FFFF
- mem rule 2_0 Address space: 0x2C00_0000 - 0x2C3F_FFFF
- mem rule 2_1 Address space: 0x2C40_0000 - 0x2C7F_FFFF
- mem rule 2_2 Address space: 0x2C80_0000 - 0x2CBF_FFFF
- mem rule 2_3 Address space: 0x2CC0_0000 - 0x2CFF_FFFF
- mem rule 2_4 Address space: 0x2D00_0000 - 0x2D3F_FFFF
- mem rule 2_5 Address space: 0x2D40_0000 - 0x2D7F_FFFF
- mem rule 2_6 Address space: 0x2D80_0000 - 0x2DBF_FFFF
- mem rule 2_7 Address space: 0x2DC0_0000 - 0x2DFF_FFFF
- mem rule 3_0 Address space: 0x2E00_0000 - 0x2E3F_FFFF
- mem rule 3_1 Address space: 0x2E40_0000 - 0x2E7F_FFFF
- mem rule 3_2 Address space: 0x2E80_0000 - 0x2EBF_FFFF
- mem rule 3_3 Address space: 0x2EC0_0000 - 0x2EFF_FFFF
- mem rule 3_4 Address space: 0x2F00_0000 - 0x2F3F_FFFF
- mem rule 3_5 Address space: 0x2F40_0000 - 0x2F7F_FFFF
- mem rule 3_6 Address space: 0x2F80_0000 - 0x2FBF_FFFF
- mem rule 3_7 Address space: 0x2FC0_0000 - 0x2FFF_FFFF

Table 988. Offset

Register	Offset
EC_CTRL_SPIFI_MEM_RULE0	C0h
EC_CTRL_SPIFI_MEM_RULE1	C4h
EC_CTRL_SPIFI_MEM_RULE2	C8h
EC_CTRL_SPIFI_MEM_RULE3	CCh

Table 989. Security access rules for SPIFI memory

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved	-
29:28	RULE7	R/W	secure control rule7 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
27:26	reserved	-	reserved	-
25:24	RULE6	R/W	secure control rule6 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
23:22	reserved	-	reserved	-
21:20	RULE5	R/W	secure control rule5 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
19:18	reserved	-	reserved	-
17:16	RULE4	R/W	secure control rule4 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
15:14	reserved	-	reserved	-
13:12	RULE3	R/W	secure control rule3 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
11:10	reserved	-	reserved	-

Table 989. Security access rules for SPIFI memory...continued

Bit	Name	Access	Description and value	Reset value
9:8	RULE2	R/W	secure control rule2 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
7:6	reserved	-	reserved	-
5:4	RULE1	R/W	secure control rule1 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
3:2	reserved	-	reserved	-
1:0	RULE0	R/W	secure control rule0 Can be set when the write_lock of the check_reg is '0'.	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.14 Security access rules for both APB bridge slaves (SEC_CTRL_APB_BRIDGE_SLAVE_RULE)

Table 990. Security access rules for both APB bridge slaves

- Offset: D0h

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved	-
5:4	APBBRIDGE1_RULE	R/W	security access rules for the whole APB bridge 1	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
3:2	reserved	-	reserved	-
1:0	APBBRIDGE0_RULE	R/W	security access rules for the whole APB bridge 0	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.15 Security access rules for APB bridge 0 peripherals (SEC_CTRL_APB_BRIDGE0_MEM_CTRL0)

Table 991. Security access rules for APB bridge 0 peripherals - Offset: E0h

Bit	Name	Access	Description and value	Reset value
31:26	reserved	-	reserved	-
25:24	INPUTMUX_RULE	R/W	peripheral input multiplexing 00b non-secure and non-privilege user access allowed 01b non-secure and privilege access allowed 10b secure and non-privilege user access allowed 11b secure and privilege user access allowed	0
23:22	reserved	-	reserved	-
21:20	SEC_PINT_RULE	R/W	secure pin interrupt and pattern match 00b non-secure and non-privilege user access allowed 01b non-secure and privilege access allowed 10b secure and non-privilege user access allowed 11b secure and privilege user access allowed	0
19:18	reserved	-	reserved	-
17:16	PINT_RULE	R/W	pin interrupt and pattern match 00b non-secure and non-privilege user access allowed 01b non-secure and privilege access allowed 10b secure and non-privilege user access allowed 11b secure and privilege user access allowed	0
15:10	reserved	-	reserved	-

Table 991. Security access rules for APB bridge 0 peripherals - Offset: E0h...continued

Bit	Name	Access	Description and value	Reset value
9:8	GINT0_RULE	R/W	GPIO input interrupt 0	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
7:6	reserved	-	reserved	-
5:4	IOCON_RULE	R/W	I/O configuration	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
3:2	reserved	-	reserved	-
1:0	SYSCON_RULE	R/W	system configuration	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.16 Security access rules for APB bridge 0 peripherals (SEC_CTRL_APB_BRIDGE0_MEM_CTRL1)

Table 992. Security access rules for APB bridge 0 peripherals - Offset: E4h

Bit	Name	Access	Description and value	Reset value
31:26	reserved	-	reserved	-
25:24	UTICK_RULE	R/W	micro-timer	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
23:22	reserved	-	reserved	-
21:20	MRT_RULE	R/W	multirate timer	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
19:18	reserved	-	reserved	-
17:16	WWDT_RULE	R/W	windowed watchdog timer	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
15:6	reserved	-	reserved	-
5:4	CTIMER1_RULE	R/W	standard counter/timer 1	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
3:2	reserved	-	reserved	-
1:0	CTIMER0_RULE	R/W	standard counter/timer 0	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.17 Security access rules for APB bridge 0 peripherals (SEC_CTRL_APB_BRIDGE0_MEM_CTRL2)

Table 993. Security access rules for APB bridge 0 peripherals - Offset: E8h

Bit	Name	Access	Description and value	Reset value
31:14	reserved	-	reserved	-
13:12	ANACTRL_RULE	R/W	analog modules controller	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
11:0	reserved	-	reserved	-

45.5.14.2.18 Security access rules for APB bridge 0 peripherals (SEC_CTRL_APB_BRIDGE0_MEM_CTRL3)

Table 994. Security access rules for APB Bridge 0 peripherals - Offset: ECh

Bit	Name	Access	Description and value	Reset value
31:26	reserved	-	reserved	-
25:24	EZH_RULE	R/W	EZH slave interface	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
23:18	reserved	-	reserved	-
25:18	reserved	-	reserved	-
17:16	SCTIPU_RULE	R/W	SCTIPU slave interface	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
15:0	reserved	-	reserved	-

45.5.14.2.19 Security access rules for APB bridge 1 peripherals (SEC_CTRL_APB_BRIDGE1_MEM_CTRL0)

Table 995. Security access rules for APB bridge 1 peripherals - Offset: F0h

Bit	Name	Access	Description and value	Reset value
31:14	reserved	-	reserved	-
13:12	SYSCTRL_RULE	R/W	system controller	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
11:2	reserved	-	reserved	-
1:0	PMC_RULE	R/W	power manager controller	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.20 Security access rules for APB bridge 1 peripherals (SEC_CTRL_APB_BRIDGE1_MEM_CTRL1)

Table 996. Security access rules for APB bridge 1 peripherals - Offset: F4h

Bit	Name	Access	Description and value	Reset value
31:22	reserved	-	reserved	-
21:20	OSEVENT_RULE	R/W	OS event timer	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
19:18	reserved	-	reserved	-
17:16	RTC_RULE	R/W	real-time counter	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
15:10	reserved	-	reserved	-

Table 996. Security access rules for APB bridge 1 peripherals - Offset: F4h...continued

Bit	Name	Access	Description and value		Reset value
9:8	CTIMER4_RULE	R/W	standard counter/timer 4		0
			00b	non-secure and non-privilege user access allowed	
			01b	non-secure and privilege access allowed	
			10b	secure and non-privilege user access allowed	
			11b	secure and privilege user access allowed	
7:6	reserved	-	reserved		-
5:4	CTIMER3_RULE	R/W	standard counter/timer 3		0
			00b	non-secure and non-privilege user access allowed	
			01b	non-secure and privilege access allowed	
			10b	secure and non-privilege user access allowed	
			11b	secure and privilege user access allowed	
3:2	reserved	-	reserved		-
1:0	CTIMER2_RULE	R/W	standard counter/timer 2		0
			00b	non-secure and non-privilege user access allowed	
			01b	non-secure and privilege access allowed	
			10b	secure and non-privilege user access allowed	
			11b	secure and privilege user access allowed	

45.5.14.2.21 Security access rules for APB Bridge 1 peripherals (SEC_CTRL_APB_BRIDGE1_MEM_CTRL2)

Table 997. Security access rules for APB Bridge 1 peripherals - Offset: F8h

Bit	Name	Access	Description and value		Reset value
31:22	reserved	-	reserved		-
21:20	PRINCE_RULE	R/W	prince/NVM_NXP_CTRL		0
			00b	non-secure and non-privilege user access allowed	
			01b	non-secure and privilege access allowed	
			10b	secure and non-privilege user access allowed	
			11b	secure and privilege user access allowed	
19:18	reserved	-	reserved		-

Table 997. Security access rules for APB Bridge 1 peripherals - Offset: F8h...continued

Bit	Name	Access	Description and value	Reset value
17:16	FLASH_CTRL_RULE	R/W	flash controller	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
15:0	reserved	-	reserved	-

45.5.14.2.22 Security access rules for APB bridge 1 peripherals (SEC_CTRL_APB_BRIDGE1_MEM_CTRL3)

Table 998. Security access rules for APB bridge 1 peripherals - Offset: FCh

Bit	Name	Access	Description and value	Reset value
31:26	reserved	-	reserved	-
25:24	ROMCP_RULE	R/W	ROM patch controller	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
23:22	reserved	-	reserved	-
25:22	reserved	-	reserved	-
21:20	PLU_RULE	R/W	programmable look-up logic	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
19:14	reserved	-	reserved	-
13:12	PUF_RULE	R/W	PUF	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
11:10	reserved	-	reserved	-

Table 998. Security access rules for APB bridge 1 peripherals - Offset: FCh...continued

Bit	Name	Access	Description and value		Reset value
9:8	RNG_RULE	R/W	true random number generator		0
			00b	non-secure and non-privilege user access allowed	
			01b	non-secure and privilege access allowed	
			10b	secure and non-privilege user access allowed	
			11b	secure and privilege user access allowed	
7:0	reserved	-	reserved		-

45.5.14.2.23 Security access rules for AHB peripherals on port 7 - slave 0 (SEC_CTRL_AHB_PORT7_SLAVE0_RULE)

Table 999. Security access rules for AHB peripherals on port 7 - slave 0 - Offset: 100h

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved	-
29:28	FLEXCOMM1_RULE	R/W	Flexcomm interface 1	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
27:26	reserved	-	reserved	-
25:24	FLEXCOMM0_RULE	R/W	Flexcomm interface 0	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
23:22	reserved	-	reserved	-
21:20	SCT_RULE	R/W	SCTimer	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
19:10	reserved	-	reserved	-

Table 999. Security access rules for AHB peripherals on port 7 - slave 0 - Offset: 100h...continued

Bit	Name	Access	Description and value	Reset value
9:8	DMA0_RULE	R/W	DMA controller	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
7:0	reserved	-	reserved	-

45.5.14.2.24 Security access rules for AHB peripherals on port 7 - slave 1 (SEC_CTRL_AHB_PORT7_SLAVE1_RULE)

Table 1000. Security access rules for AHB peripherals on port 7 - slave 1 - Offset: 104h

Bit	Name	Access	Description and value	Reset value
31:18	reserved	-	reserved	-
17:16	GPIO0_RULE	R/W	high-speed GPIO	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
15:2	reserved	-	reserved	-
1:0	FLEXCOMM2_RULE	R/W	Flexcomm interface 2	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.25 Security access rules for AHB peripherals on port 8 - slave 0 (SEC_CTRL_AHB_PORT8_SLAVE0_RULE)

Table 1001. Security access rules for AHB peripherals on port 8 - slave 0 - Offset: 110h

Bit	Name	Access	Description and value	Reset value
31:22	reserved	-	reserved	-
21:20	CRC_RULE	R/W	CRC engine	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
19:0	reserved	-	reserved	-

45.5.14.2.26 Security access rules for AHB peripherals on port 8 - slave 1 (SEC_CTRL_AHB_PORT8_SLAVE1_RULE)

Table 1002. Security access rules for AHB peripherals on port 8 - slave 1 - Offset: 114h

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved	-
29:28	SPIFI_RULE	R/W	SPIFI	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
27:18	reserved	-	reserved	-
17:16	DBG_MAILBOX_RULE	R/W	debug mailbox (aka ISP-AP)	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
15:0	reserved	-	reserved	-

45.5.14.2.27 Security access rules for AHB peripherals on port 9 - slave 0 (SEC_CTRL_AHB_PORT9_SLAVE0_RULE)

Table 1003. Security access rules for AHB peripherals on port 9 - slave 0 - Offset: 120h

Bit	Name	Access	Description and value	Reset value
31:30	reserved	-	reserved	-
29:28	DMA1_RULE	R/W	DMA controller (secure)	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
27:22	reserved	-	reserved	-
21:20	CASPER_RULE	R/W	RSA/ECC crypto accelerator	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
19:18	reserved	-	reserved	-
17:16	HASH_RULE	R/W	SHA-2 crypto registers	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
15:6	reserved	-	reserved	-
5:4	CDOG_RULE	R/W	code watchdog registers	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
3:0	reserved	-	reserved	-
5:0	reserved	-	reserved	-

45.5.14.2.28 Security access rules for AHB peripherals on port 9 - slave 1 (SEC_CTRL_AHB_PORT9_SLAVE1_RULE)

Table 1004. Security access rules for AHB peripherals on port 9 - slave 1 - Offset: 124h

Bit	Name	Access	Description and value	Reset value
31:6	reserved	-	reserved	-
5:4	AHB_SEC_CTRL_RULE	R/W	AHB secure controller	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
3:2	reserved	-	reserved	-
1:0	GPIO1_RULE	R/W	secure high-speed GPIO	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.29 Security access rules for AHB_SEC_CTRL memory (SEC_CTRL_AHB_SEC_CTRL_MEM_RULE)

Table 1005. Security access rules for AHB_SEC_CTRL memory - Offset: 130h

Bit	Name	Access	Description and value	Reset value
31:14	reserved	-	reserved	-
13:12	AHB_SEC_CTRL_SECT_3_RULE	R/W	address space: 0x400A_F000 - 0x400A_FFFF	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
11:10	reserved	-	reserved	-

Table 1005. Security access rules for AHB_SEC_CTRL memory - Offset: 130h...continued

Bit	Name	Access	Description and value		Reset value
9:8	AHB_SEC_CTRL_SECT_2_RULE	R/W	address space: 0x400A_E000 - 0x400A_EFFF		0
			00b	non-secure and non-privilege user access allowed	
			01b	non-secure and privilege access allowed	
			10b	secure and non-privilege user access allowed	
			11b	secure and privilege user access allowed	
7:6	reserved	-	reserved		-
5:4	AHB_SEC_CTRL_SECT_1_RULE	R/W	address space: 0x400A_D000 - 0x400A_DFFF		0
			00b	non-secure and non-privilege user access allowed	
			01b	non-secure and privilege access allowed	
			10b	secure and non-privilege user access allowed	
			11b	secure and privilege user access allowed	
3:2	reserved	-	reserved		-
1:0	AHB_SEC_CTRL_SECT_0_RULE	R/W	address space: 0x400A_0000 - 0x400A_CFFF		0
			00b	non-secure and non-privilege user access allowed	
			01b	non-secure and privilege access allowed	
			10b	secure and non-privilege user access allowed	
			11b	secure and privilege user access allowed	

45.5.14.2.30 Security access rules for AHB peripherals on port 10 - slave 0 (SEC_CTRL_AHB_PORT10_SLAVE0_RULE)

Table 1006. Security access rules for AHB peripherals on port 10 - slave 0 - Offset: 140h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1:0	BLE_RADIO_RULE	R/W	Bluetooth Low Energy controller	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	

45.5.14.2.31 Most recent security violation address for AHB layer n (SEC_VIO_ADDRa)

For a = 0 to 9

Table 1007. Most recent security violation address for AHB layer n - Offset: E00h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	SEC_VIO_ADDR	R	security violation address for AHB layer	0

45.5.14.2.32 Most recent security violation miscellaneous information for AHB layer n (SEC_VIO_MISC_INFOa)

For a = 0 to 9

Table 1008. Most recent security violation miscellaneous information for AHB layer n - Offset: E80h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:12	reserved	-	reserved	-
11:8	SEC_VIO_INFO_MASTER	R	security violation master number	0
			0000b CPU0 code	
			0001b CPU0 system	
			0010b reserved	
			0011b reserved	
			0100b reserved	
			0101b SDMA0	
			1000b reserved	
			1001b reserved	
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	R	bit [5:4]: master sec level and privilege level bit [7:6]: anti-pol value for master sec level and privilege level	0
3:2	reserved	-	reserved	-
1	SEC_VIO_INFO_DATA_ACCESS	R	security violation access data/code indicator	0
			0b code access	
			1b data access	
0	SEC_VIO_INFO_WRITE	R	security violation access read/write indicator	0
			0b read access	
			1b write access	

Table 1009. Most recent security violation miscellaneous information for AHB layer n - Offset: E80h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:12	reserved	-	reserved	-
11:8	SEC_VIO_INFO_MASTER	R	security violation master number	0
			0000b CPU0 code	
			0001b CPU0 system	
			0010b reserved	
			0011b reserved	
			0100b reserved	

Table 1009. Most recent security violation miscellaneous information for AHB layer n - Offset: E80h + (a × 4h)...*continued*

Bit	Name	Access	Description and value		Reset value
			0101b	SDMA0	
			0110b	EZH-B data	
			0111b	EZH-B instruction	
			1000b	reserved	
7:4	SEC_VIO_INFO_MASTER_SEC_LEVEL	R	bit [5:4]: master sec level and privilege level bit [7:6]: anti-pol value for master sec level and privilege level		0
3:2	reserved	-	reserved		-
1	SEC_VIO_INFO_DATA_ACCESS	R	security violation access data/code indicator		0
			0b	code access	
			1b	data access	
0	SEC_VIO_INFO_WRITE	R	security violation access read/write indicator		0
			0b	read access	
			1b	write access	

45.5.14.2.33 Security violation address/information registers valid flags (SEC_VIO_INFO_VALID)

Table 1010. Security violation address/information registers valid flags - Offset: F00h

Bit	Name	Access	Description and value	Reset value
31:11	reserved	-	reserved	-
10	VIO_INFO_VALID10	R/W	violation information valid flag for AHB port 10 Write 1 to clear.	0
			0b not valid	
			1b valid (violation occurred)	
9	VIO_INFO_VALID9	R/W	violation information valid flag for AHB port 9 Write 1 to clear.	0
			0b not valid	
			1b valid (violation occurred)	
8	VIO_INFO_VALID8	R/W	violation information valid flag for AHB port 8 Write 1 to clear.	0
			0b not valid	
			1b valid (violation occurred)	
7	VIO_INFO_VALID7	R/W	violation information valid flag for AHB port 7 Write 1 to clear.	0
			0b not valid	
			1b valid (violation occurred)	
6	VIO_INFO_VALID6	R/W	violation information valid flag for AHB port 6 Write 1 to clear.	0
			0b not valid	
			1b valid (violation occurred)	
5	VIO_INFO_VALID5	R/W	violation information valid flag for AHB port 5 Write 1 to clear.	0
			0b not valid	
			1b valid (violation occurred)	
4	VIO_INFO_VALID4	R/W	violation information valid flag for AHB port 4 Write 1 to clear.	0
			0b not valid	
			1b valid (violation occurred)	
3	VIO_INFO_VALID3	R/W	violation information valid flag for AHB port 3 Write 1 to clear.	0
			0b not valid	
			1b valid (violation occurred)	

Table 1010. Security violation address/information registers valid flags - Offset: F00h...continued

Bit	Name	Access	Description and value	Reset value
2	VIO_INFO_VALID2	R/W	violation information valid flag for AHB port 2 Write 1 to clear.	0
			0b not valid	
			1b valid (violation occurred)	
1	VIO_INFO_VALID1	R/W	violation information valid flag for AHB port 1 Write 1 to clear.	0
			0b not valid	
			1b valid (violation occurred)	
0	VIO_INFO_VALID0	R/W	violation information valid flag for AHB port 1 Write 1 to clear.	0
			0b not valid	
			1b valid (violation occurred)	

45.5.14.2.34 Secure GPIO mask for port 0 pins (SEC_GPIO_MASK0)

This register is used to block leakage of secure interface (GPIOs, I2C, UART configured as secure peripherals) pin states to the non-secure world through non-secure GPIO port control registers.

Table 1011. Secure GPIO mask for port 0 pins - Offset: F80h

Bit	Name	Access	Description and value	Reset value
31:23	reserved	-	reserved	-
22	PIO0_PIN22_SEC_MASK	R/W	secure mask for pin P0_22	1
			0b pin state is blocked to non-secure world	
			1b pin state is readable by non-secure world	
21	PIO0_PIN21_SEC_MASK	R/W	secure mask for pin P0_21	1
			0b pin state is blocked to non-secure world	
			1b pin state is readable by non-secure world	
20	PIO0_PIN20_SEC_MASK	R/W	secure mask for pin P0_20	1
			0b pin state is blocked to non-secure world	
			1b pin state is readable by non-secure world	
19	PIO0_PIN19_SEC_MASK	R/W	secure mask for pin P0_19	1
			0b pin state is blocked to non-secure world	
			1b pin state is readable by non-secure world	

Table 1011. Secure GPIO mask for port 0 pins - Offset: F80h...continued

Bit	Name	Access	Description and value		Reset value
18	PIO0_PIN18_SEC_MASK	R/W	secure mask for pin P0_18		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	
17	PIO0_PIN17_SEC_MASK	R/W	secure mask for pin P0_17		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	
16	PIO0_PIN16_SEC_MASK	R/W	secure mask for pin P0_16		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	
15	PIO0_PIN15_SEC_MASK	R/W	secure mask for pin P0_15		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	
14	PIO0_PIN14_SEC_MASK	R/W	secure mask for pin P0_14		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	
13	PIO0_PIN13_SEC_MASK	R/W	secure mask for pin P0_13		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	
12	PIO0_PIN12_SEC_MASK	R/W	secure mask for pin P0_12		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	
11	PIO0_PIN11_SEC_MASK	R/W	secure mask for pin P0_11		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	

Table 1011. Secure GPIO mask for port 0 pins - Offset: F80h...continued

Bit	Name	Access	Description and value		Reset value
10	PIO0_PIN10_SEC_MASK	R/W	secure mask for pin P0_10		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	
9	PIO0_PIN9_SEC_MASK	R/W	secure mask for pin P0_9		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	
8	PIO0_PIN8_SEC_MASK	R/W	secure mask for pin P0_8		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	
7	PIO0_PIN7_SEC_MASK	R/W	secure mask for pin P0_7		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	
6	PIO0_PIN6_SEC_MASK	R/W	secure mask for pin P0_6		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	
5	PIO0_PIN5_SEC_MASK	R/W	secure mask for pin P0_5		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	
4	PIO0_PIN4_SEC_MASK	R/W	secure mask for pin P0_4		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	
3	PIO0_PIN3_SEC_MASK	R/W	secure mask for pin P0_3		1
			0b	pin state is blocked to non-secure world	
			1b	pin state is readable by non-secure world	

Table 1011. Secure GPIO mask for port 0 pins - Offset: F80h...continued

Bit	Name	Access	Description and value	Reset value
2	PIO0_PIN2_SEC_MASK	R/W	secure mask for pin P0_2	1
			0b pin state is blocked to non-secure world	
			1b pin state is readable by non-secure world	
1	PIO0_PIN1_SEC_MASK	R/W	secure mask for pin P0_1	1
			0b pin state is blocked to non-secure world	
			1b pin state is readable by non-secure world	
0	PIO0_PIN0_SEC_MASK	R/W	secure mask for pin P0_0	1
			0b pin state is blocked to non-secure world	
			1b pin state is readable by non-secure world	

45.5.14.2.35 Security general-purpose register access control (SEC_MASK_LOCK)

Table 1012. Security general-purpose register access control - Offset: FBCh

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1:0	SEC_GPIO_MASK0_LOCK	R/W	SEC_GPIO_MASK0 register write-lock 2'b10: SEC_GPIO_MASK0 can be written.	1
			01b restricted mode	
			10b writable	

45.5.14.2.36 Master secure level register (MASTER_SEC_LEVEL)

Table 1013. Master secure level register - Offset: FD0h

Bit	Name	Access	Description and value	Reset value
31:30	MASTER_SEC_LEVEL_LOCK	R/W	MASTER_SEC_LEVEL write-lock When 2'b10, this register can be written. With any other value, this register cannot be written (including this bit field).	1
			01b restricted mode	
			10b writable	
29:26	reserved	-	reserved Read value is undefined, only zero must be written.	-

Table 1013. Master secure level register - Offset: FD0h...continued

Bit	Name	Access	Description and value		Reset value
25:24	SDMA1	R/W	system DMA 1 security level		0
			00b	non-secure and non-privilege user access allowed	
			01b	non-secure and privilege access allowed	
			10b	secure and non-privilege user access allowed	
			11b	secure and privilege user access allowed	
23:22	reserved	-	reserved Read value is undefined, only zero must be written.		-
21:20	HASH	R/W	hash		0
			00b	non-secure and non-privilege user access allowed	
			01b	non-secure and privilege access allowed	
			10b	secure and non-privilege user access allowed	
			11b	secure and privilege user access allowed	
19:16	reserved	-	reserved Read value is undefined, only zero must be written.		-
15:14	EZH_I	R/W	EZH instruction		0
			00b	non-secure and non-privilege user access allowed	
			01b	non-secure and privilege access allowed	
			10b	secure and non-privilege user access allowed	
			11b	secure and privilege user access allowed	
13:12	EZH_D	R/W	EZH data		0
			00b	non-secure and non-privilege user access allowed	
			01b	non-secure and privilege access allowed	
			10b	secure and non-privilege user access allowed	
			11b	secure and privilege user access allowed	

Table 1013. Master secure level register - Offset: FD0h...continued

Bit	Name	Access	Description and value	Reset value
19:12	reserved	-	reserved	-
11:10	SDMA0	R/W	system DMA 0	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
9:0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.5.14.2.37 Master secure level antipole register (MASTER_SEC_ANTI_POL_REG)

Table 1014. Master secure level anti-pole register - Offset: FD4h

Bit	Name	Access	Description and value	Reset value
31:30	MASTER_SEC_LEVEL_ANTI_POL_LOCK	R/W	MASTER_SEC_ANTI_POL_LEVEL register write-lock When 2'b10, this register can be written. With any other value, this register cannot be written (including this bit field).	1
			01b restricted mode	
			10b writable	
29:26	reserved	-	reserved Read value is undefined, only zero must be written.	-
25:24	SDMA1	R/W	system DMA 1 security level Must be equal to NOT(MASTER_SEC_LEVEL.SDMA1).	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
23:22	reserved	-	reserved Read value is undefined, only zero must be written.	-
21:20	HASH	R/W	hash Must be equal to NOT(MASTER_SEC_LEVEL.HASH).	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
19:12	reserved	-	reserved Read value is undefined, only zero must be written.	-

Table 1014. Master secure level anti-pole register - Offset: FD4h...continued

Bit	Name	Access	Description and value	Reset value
15:14	EZH_I	R/W	EZH instruction Must be equal to NOT(MASTER_SEC_LEVEL.EZH_I).	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
13:12	EZH_D	R/W	EZH data Must be equal to NOT(MASTER_SEC_LEVEL.EZH_D).	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
11:10	SDMA0	R/W	system DMA 0 Must be equal to NOT(MASTER_SEC_LEVEL.SDMA0).	0
			00b non-secure and non-privilege user access allowed	
			01b non-secure and privilege access allowed	
			10b secure and non-privilege user access allowed	
			11b secure and privilege user access allowed	
9:0	reserved	-	reserved Read value is undefined, only zero must be written.	-

45.5.14.2.38 Miscellaneous control signals for in Cortex M33 (CPU0) (CPU0_LOCK_REG)

Table 1015. Miscellaneous control signals for in Cortex M33 (CPU0) - Offset: FECh

Bit	Name	Access	Description and value	Reset value
31:30	CPU0_LOCK_REG_LOCK	R/W	CPU0_LOCK_REG write-lock When 2'b10: This register can be written.	1

Table 1015. Miscellaneous control signals for in Cortex M33 (CPU0) - Offset: FECh...continued

Bit	Name	Access	Description and value	Reset value
			All other values: This register cannot be written (including this bit field).	
			01b restricted mode	
			10b writable	
29:10	reserved	-	reserved	-
9:8	LOCK_SAU	R/W	Cortex M33 (CPU0) SAU registers write-lock 2'b10: Cortex M33 (CPU0) LOCKSAU is 0. Enable writes to SAU_CTRL, SAU_RNR, SAU_RBAR and SAU_RLAR registers. All other values: Cortex M33 (CPU0) LOCKSAU is 1. Disable writes to SAU_CTRL, SAU_RNR, SAU_RBAR and SAU_RLAR registers.	1
			01b restricted mode	
			10b writable	
7:6	LOCK_S_MPU	R/W	Cortex M33 (CPU0) secure MPU registers write-lock 2'b10: Cortex M33 (CPU0) LOCKSVTAIRCR is 0. Enable writes to the VTOR_S, AIRCR.PRIS, ICR.BFHFNMINS registers. All other values: Cortex M33 (CPU0) LOCKSVTAIRCR is 1. Disable writes to the VTOR_S, AIRCR.PRIS, ICR.BFHFNMINS registers.	1
			01b restricted mode	
			10b writable	
5:4	LOCK_S_VTAIRCR	R/W	Cortex M33 (CPU0) VTOR_S, AIRCR.PRIS, ICR.BFHFNMINS registers write-lock 2'b10: Cortex M33 (CPU0) LOCKSVTAIRCR is 0. Enable writes to the VTOR_S, AIRCR.PRIS, ICR.BFHFNMINS registers. All other values: Cortex M33 (CPU0) LOCKSVTAIRCR is 1. Disable writes to the VTOR_S, AIRCR.PRIS, ICR.BFHFNMINS registers.	1
			01b restricted mode	
			10b writable	
3:2	LOCK_NS_MPU	R/W	Cortex M33 (CPU0) non-secure MPU register write-lock 2'b10: Cortex M33 (CPU0) LOCKNSMPU is 0. Enable writes to non-secure MPU registers. All other values: Cortex M33 (CPU0) LOCKNSMPU is 1. Disable writes to non-secure MPU registers.	1
			01b restricted mode	
			10b writable	
1:0	LOCL_NS_VTOR	R/W	Cortex M33 (CPU0) VTOR_NS register write-lock	1

Table 1015. Miscalleneous control signals for in Cortex M33 (CPU0) - Offset: FECh...continued

Bit	Name	Access	Description and value		Reset value
			2'b10: Cortex M33 (CPU0) LOCKNSVTOR is 0. Enable writes to the VTOR_NS register. All other values: Cortex M33 (CPU0) LOCKNSVTOR is 1. Disable writes to the VTOR_NS register.		
			01b	restricted mode	
			10b	writable	

45.5.14.2.39 Secure control duplicate register (MISC_CTRL_DP_REG)

Table 1016. Secure control duplicate register - Offset: FF8h

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved	-
15:14	IDAU_ALL_NS	R/W	disable IDAU 00, 11, 10: IDAU is enabled (restricted mode). 01: IDAU is disabled, so all memories are attributed as non-secure memory.	1
			01b IDAU is disabled	
			10b IDAU is enabled	
13:12	DISABLE_SMART_MASTER_STRICT_MODE	R/W	disable smart master strict mode 00, 11, 10: Smart masters in strict mode. Can execute, read and write to memories at same level only (Mode recommended by ARM)(restrictive mode). 01: 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. It applies to CPU.	1
			01b smart master in tier mode	
			10b smart master in strict mode	
11:10	DISABLE_SIMPLE_MASTER_STRICT_MODE	R/W	disable simple master strict mode 00, 11, 10: Simple master in strict mode. Can read and write to memories at same level only (Mode recommended by ARM) (restrictive mode). 01: Simple master in tier mode. Can read and write to memories at same or below level. It applies to DMA0 and DMA1.	1
			01b simple master in tier mode	
			10b simple master in strict mode	
9:8	DISABLE_VIOLATION_ABORT	R/W	disable secure violation abort 10: The violation detected by the secure checker causes abort. All other values: The violation detected by the secure checker does not cause abort. However, secure_violation_irq is still asserted and serviced by ISR.	1
			01b disable abort for secure checker	
			10b enable abort for secure checker	
7:6	ENABLE_NS_PRIV_CHECK	R/W	enable non-secure privilege check for AHB matrix 10: Disabled. All other values: Enabled (restrictive mode).	1
			01b restricted mode	
			10b disable check	

Table 1016. Secure control duplicate register - Offset: FF8h...continued

Bit	Name	Access	Description and value	Reset value
5:4	ENABLE_S_PRIV_CHECK	R/W	enable secure privilege check for AHB matrix 10: Disabled. All other values: Enabled (restrictive mode).	1
			01b restricted mode	
			10b disable check	
3:2	ENABLE_SECURE_CHECKING	R/W	enable secure check for AHB matrix 10: Disabled. All other values: Enabled (restrictive mode).	1
			01b restricted mode	
			10b disable check	
1:0	WRITE_LOCK	R/W	write lock 10: secure_ctrl_group_rule registers and this register itself can be written. All other values: secure_ctrl_group_rule registers and this register itself cannot be written (restricted mode). If this field does not match misc_ctrl_dp_reg[31:30], misc_ctrl_reg and misc_ctrl_dp_reg and all secure_ctrl_group_rule registers are write-locked. When the control fields below do not match the corresponding fields in the duplicate register, the related control signals are set to the restrictive mode.	1
			01b restricted mode	
			10b secure control registers can be written	

45.5.14.2.40 Secure control register (MISC_CTRL_REG)

Table 1017. Secure control register - Offset: FFCh

Bit	Name	Access	Description and value	Reset value
31:16	reserved	-	reserved	-
15:14	IDAU_ALL_NS	R/W	disable IDAU 00, 11, 10: IDAU is enabled (restrictive mode). 01: IDAU is disabled, so all memories are attributed as non-secure memory.	1
			01b IDAU is disabled	
			10b IDAU is enabled	

Table 1017. Secure control register - Offset: FFCh...continued

Bit	Name	Access	Description and value	Reset value
13:12	DISABLE_SMART_MASTER_STRICT_MODE	R/W	disable smart master strict mode 00, 11, 10: Smart masters in strict mode. Can execute, read and write to memories at same level only (Mode recommended by ARM) (restrictive mode). 01: Smart masters in tier mode. Can execute at same level only, but read and write to memories at same or below level. Signaling is passed through to bus matrix. It applies to DSP.	1
			01b smart master in tier mode	
			10b smart master in strict mode	
11:10	DISABLE_SIMPLE_MASTER_STRICT_MODE	R/W	disable simple master strict mode 00, 11, 10: Simple master in strict mode. Can read and write to memories at same level only (Mode recommended by ARM) (restrictive mode). 01: Simple master in tier mode. Can read and write to memories at same or below level. It applies to DMA0 and DMA1.	1
			01b simple master in tier mode	
			10b simple master in strict mode	
9:8	DISABLE_VIOLATION_ABORT	R/W	disable secure violation abort 10: The violation detected by the secure checker causes abort. All other values: The violation detected by the secure checker does not cause abort. However, secure_violation_irq is still asserted and serviced by ISR.	1
			01b disable abort for secure checker	
			10b enable abort for secure checker	
7:6	ENABLE_NS_PRIV_CHECK	R/W	enable non-secure privilege check for AHB matrix 10: Disabled. All other values: Enabled (restrictive mode).	1
			01b enabled (restricted mode)	
			10b disable check	
5:4	ENABLE_S_PRIV_CHECK	R/W	enable secure privilege check for AHB matrix 10: Disabled. All other values: Enabled (restrictive mode).	1
			01b enabled (restricted mode)	
			10b disable check	

Table 1017. Secure control register - Offset: FFCh...continued

Bit	Name	Access	Description and value	Reset value
3:2	ENABLE_SECURE_CHECKING	R/W	enable secure check for AHB matrix 10: Disabled. All other values: Enabled (restrictive mode).	1
			01b enabled (restricted mode)	
			10b disable check	
1:0	WRITE_LOCK	R/W	write lock 10: secure_ctrl_group_rule registers and this register itself can be written. All other values: secure_ctrl_group_rule registers and this register itself cannot be written (restricted mode). If this field does not match misc_ctrl_dp_reg[31:30], misc_ctrl_reg and misc_ctrl_dp_reg and all secure_ctrl_group_rule registers are write-locked. When the control fields below do not match the corresponding fields in the duplicate register, the related control signals are set to the restrictive mode.	1
			01b restricted mode	
			10b secure control registers can be written	

45.5.15 Bluetooth Low Energy

No register map is available for the Bluetooth Low Energy radio. A complete API description to use the Bluetooth Low Energy module can be found in the section [Section 44](#).

45.6 Arm NVIC and TrustZone

The Arm CM33 contains specific registers to control the NVIC, SAU, and SCB. [Table 1018](#) gives an overview of the base addresses for these registers.

Table 1018. Arm NVIC and TrustZone

Arm function	Base address	Reference
Nested vector interrupt control (NVIC)	E000_E100h	Section 45.6.1
System control bus (SCB)	E000_ED00h	Section 45.6.2
Secure attribute unit (SAU)	E000_ED00h	Section 45.6.3

45.6.1 Nested vector interrupt control (NVIC)

45.6.1.1 NVIC memory map

The NVIC registers are located at base address: E000_E100h.

Table 1019. Nested vector interrupt control registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
ISER0 - ISER15	R/W	0h - 3h	interrupt set enable register	32	0000_0000h	Section 45.6.1.2.1
ICER0 - ICER15	R/W	80h - BCh	interrupt clear enable register	32	0000_0000h	Section 45.6.1.2.2
ISPR0 - ISPR15	R/W	100h - 13Ch	interrupt set pending register	32	0000_0000h	Section 45.6.1.2.3
ICPR0 - ICPR15	R/W	180h - 1BCh	interrupt clear pending register	32	0000_0000h	Section 45.6.1.2.4
IABR0 - IABR15	R/W	200h - 23Ch	interrupt active bit register	32		Section 45.6.1.2.5
ITNS0 - ITNS15	R/W	280h - 2BCh	interrupt target non-secure register	32		Section 45.6.1.2.6
IPR0 - IPR119	R/W	300h - 4DCh	interrupt priority register	32		Section 45.6.1.2.7
STIR	W	E00h	software trigger interrupt register	32	0000_0000h	Section 45.6.1.2.8

45.6.1.2 Register descriptions

45.6.1.2.1 Interrupt set enable register (ISERa)

Enables interrupts and shows which interrupts are enabled.

In an implementation with the security extension:

- The register bits can be RAZ/WI depending on the value of NVIC_ITNS.
- These registers are not banked between security states.

For a = 0 to 15

Table 1020. Interrupt set enable register - Offset: 0h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	SETENAn	R/W	interrupt set-enable bits For SETENA[m] in NVIC_ISERn, allows interrupt 32n + m to be accessed.	0
			0b write: no effect read: interrupt 32n + m disabled	
			1b write: enable interrupt 32n + m read: interrupt 32n + m enabled	

45.6.1.2.2 Interrupt clear enable register (ICERa)

Disables interrupts and shows which interrupts are enabled.

In an implementation with the security extension:

- The register bits can be RAZ/WI from non-secure state depending on the value of NVIC_ITNS.
- These registers are not banked between security states.

For a = 0 to 15

Table 1021. Interrupt clear enable register - Offset: 80h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	CLRENAn	R/W	interrupt clear-enable bits For CLRENA[m] in NVIC_ICERn, allows interrupt 32n + m to be accessed.	0
			0b write: no effect read: interrupt 32n + m disabled	
			1b write: enable interrupt 32n + m read: interrupt 32n + m enabled	

45.6.1.2.3 Interrupt set pending register (ISPRa)

Force interrupts into the pending state and shows which interrupts are pending.

In an implementation with the security extension:

- The register bits can be RAZ/WI from non-secure state depending on the value of NVIC_ITNS.
- These registers are not banked between security states.

For a = 0 to 15

Table 1022. Interrupt set pending register - Offset: 100h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	SETPENDn	R/W	interrupt set-pending bits For SETPEND[m] in NVIC_ISPRn, allows interrupt 32n + m to be accessed.	0
			0b write: no effect read: interrupt 32n + m is not pending	
			1b write: pend interrupt 32n + m read: interrupt 32n + m pending	

45.6.1.2.4 Interrupt clear pending register (ICPRa)

Removes the pending state from interrupts and shows which interrupts are pending.

In an implementation with the security extension:

- The register bits can be RAZ/WI from non-secure state depending on the value of NVIC_ITNS.
- These registers are not banked between security states.

For a = 0 to 15

Table 1023. Interrupt clear pending register - Offset: 180h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	CLRPENDn	R/W	interrupt clear-pending bits	0
			0b write: no effect read: interrupt 32n + m is not pending	
			1b write: clear pending state of interrupt 32n + m read: interrupt 32n + m is pending	

45.6.1.2.5 Interrupt active bit register (IABRa)

Indicates the active state of each interrupt.

In an implementation with the security extension:

- The register bits can be RAZ/WI from non-secure state depending on the value of NVIC_ITNS.
- These registers are not banked between security states.

For a = 0 to 15

Table 1024. Interrupt active bit register - Offset: 200h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	ACTIVE _n	R/W	active state bits For ACTIVE[m] in NVIC_IABR _n , indicates the active state for interrupt 32n + m.	0
			0b interrupt is not active	
			1b interrupt is active	

45.6.1.2.6 Interrupt target non-secure register (ITNSa)

For each group of 32 interrupts determines whether each interrupt targets non-secure or secure state.

Otherwise, this register is RAZ/WI. In an implementation with the security extension, this register is accessible from secure state only.

For a = 0 to 15

Table 1025. Interrupt target non-secure register - Offset: 280h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:0	ITNS _n	R/W	interrupt targets non-secure bits For ITNS[m] in NVIC_ITNS _n , indicates the active state for interrupt 32n + m.	0
			0b interrupt targets secure state	
			1b interrupt targets non-secure state	

45.6.1.2.7 Interrupt priority register (IPR0 - IPR119)

Provides an 8-bit priority field for each interrupt. The register is word, halfword, and byte accessible.

For a = 0 to 119

Table 1026. Interrupt target non-secure register - Offset: 300h + (a × 4h)

Bit	Name	Access	Description and value	Reset value
31:24	PRI_3	R/W	priority level for interrupt 3	0
23:16	PRI_2	R/W	priority level for interrupt 2	0
15:8	PRI_1	R/W	priority level for interrupt 1	0
7:0	PRI_0	R/W	priority level for interrupt 0	0

45.6.1.2.8 Software trigger interrupt register (STIR)

Write to the STIR to generate an interrupt from software. When the USERSETMPEND bit in the CCR is set to 1, unprivileged software can access the STIR. Only privileged software can enable unprivileged access to the STIR.

Table 1027. Software trigger interrupt register - Offset: E00h

Bit	Name	Access	Description and value	Reset value
31:9	reserved	-	reserved	-
8:0	INTID	W	interrupt ID of the interrupt to trigger, in the range 0-479 For example, a value of 0x03 specifies interrupt IRQ3.	0

45.6.2 Arm system control block registers (SCB)

45.6.2.1 SCB memory map

The system control block (SCB) provides system implementation information and system control that includes configuration, control, and reporting of system exceptions.

The SCB registers are located at base address: E000_ED00h.

Table 1028. System control block registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
AIRCR	R/W	Ch	application interrupt and reset control register	32	FA05_0000h	Section 45.6.2.2.1
SCR	R/W	10h	SCR control of features of entry to and exit from low-power state	32	0000_0000h	Section 45.6.2.2.2
SHCSR	R/W	24h	system handler control and state register	32	0000_0000h	Section 45.6.2.2.3
NSACR	R/W	8Ch	non-secure access control register	32		Section 45.6.2.2.4

45.6.2.2 Register descriptions

45.6.2.2.1 Application interrupt and reset control register (AIRCR)

The AIRCR provides sets or returns interrupt control and reset configuration.

Table 1029. Application interrupt and reset control register - Offset: Ch

Bit	Name	Access	Description and value	Reset value
31:16	VECTKEY	R	register key; reads as 0xFA05 On writes, write 0x5FA to VECTKEY. Otherwise, the write is ignored. This bit is not banked between security states.	1
15	ENDIANNESS	R	data endianness bit This bit is not banked between security states.	0
			0b little-endian	
			1b big-endian	
14	PRIS	R/W	prioritize secure exceptions The value of this bit defines if secure exception priority boosting is enabled. This bit is not banked between security states. RW from secure state and RAZ/WI from non-secure state.	0
			0b priority ranges of secure and non-secure exceptions are identical	
			1b non-secure exceptions are deprioritized	
13	BFHFNMINs	R/W	BusFault, HardFault, and NMI non-secure enable The value of this bit defines whether BusFault and NMI exceptions are non-secure. It also defines if exceptions target the non-secure Hard Fault exception. This bit is not banked between security states. RW from secure-state and RO from non-secure state.	0
			0b BusFault, HardFault, and NMI are secure	
			1b BusFault and NMI are non-secure and exceptions can target non-secure HardFault	
12:11	reserved	-	reserved	-
10:8	PRIGROUP	R/W	interrupt priority grouping field This field determines the split of group priority from subpriority. This bit is banked between security states	0
7:4	reserved	-	reserved	-
3	SYSRESETREQS	R/W	system reset request; secure state only The value of this bit defines if the SYSRESETREQ bit is functional for non-secure use. This bit is not banked between security states. R/W from secure state and RAZ/WI from non-secure state.	0
			0b SYSRESETREQ functionality available to both security states	
			1b SYSRESETREQ functionality only available to secure state	

Table 1029. Application interrupt and reset control register - Offset: Ch...continued

Bit	Name	Access	Description and value	Reset value
2	SYSRESETREQ	R/W	system reset request This bit allows software or a debugger to request a system reset. This bit is not banked between security states. R/W if SYSRESETREQS is 0. When SYSRESETREQS is set to 1, from non-secure state, this bit acts as RAZ/WI.	0
			0b do not request a system reset	
			1b request a system reset	
1	VECTCLRACTIVE	W	reserved for debug use This bit reads as 0. When writing to the register you must write 0 to this bit. Otherwise, behavior is unpredictable. This bit is not banked between security states.	
0	reserved	-	reserved	-

45.6.2.2.2 SCR control of features of entry to and exit from low-power state (SCR)

The SCR controls features of entry to and exit from low-power state.

Table 1030. SCR control of features of entry to and exit from low-power state - Offset: 10h

Bit	Name	Access	Description and value	Reset value
31:5	reserved	-	reserved	-
4	SEVONPEND	R/W	send event on pending bit When an event or interrupt enters pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE. The processor also wakes up on execution of an SEV instruction or an external event. This bit is banked between security states.	0
			0b Only enabled interrupts or events can wake up the processor, disabled interrupts are excluded.	
			1b Enabled events and all interrupts, including disabled interrupts, can wake up the processor.	
3	SLEEPDEEPS	R/W	controls if the SLEEPDEEP bit is only accessible from the secure state This bit is only accessible from the secure state. It behaves as RAZ/WI when accessed from the Non-secure state. This bit is not banked between security states.	0
			0b SLEEPDEEP bit accessible from both security states	
			1b SLEEPDEEP bit behaves as RAZ/WI when accessed from the non-secure state	
2	SLEEPDEEP	R/W	controls if the processor uses sleep or deep sleep as its low-power mode This bit is not banked between security states.	0
			0b sleep	
			1b deep sleep	

Table 1030. SCR control of features of entry to and exit from low-power state - Offset: 10h...continued

Bit	Name	Access	Description and value		Reset value
1	SLEEPONEXIT	R/W	indicates sleep-on-exit when returning from handler mode to thread mode Setting this bit to 1 enables an interrupt driven application to avoid returning to an empty main application. This bit is banked between security states.		0
			0b	do not sleep when returning to Thread mode	
			1b	enter sleep or deep sleep on return from an ISR	
10:8	PRIGROUP	R/W	interrupt priority grouping field This field determines the split of group priority from subpriority. This bit is banked between security states		0
0	reserved	-	reserved		-

45.6.2.2.3 System handler control and state register (SHCSR)

The SHCSR enables the system handlers. It indicates the pending status of the BusFault, MemManage fault, and SVC exceptions. It also indicates the active status of the system handlers.

Table 1031. System handler control and state register - Offset: 24h

Bit	Name	Access	Description and value		Reset value
31:22	reserved	-	reserved		-
21	HARDFaultPENDED	R/W	HardFault exception pended state Set to 1 to allow exception modification. This bit is banked between security states. The non-secure HardFault exception does not preempt if AIRCR.BFHFNMINS is set to zero.		0
			0b	HardFault exception modification is disabled	
			1b	HardFault exception modification is enabled	
20	SECUREFaultPENDED	R/W	SecureFault exception pended state bit Set to 1 to allow exception modification. This bit is not banked between security states.		0
			0b	SecureFault exception modification is disabled	
			1b	SecureFault exception modification is enabled	
19	SECUREFaultENA	R/W	SecureFault exception enable Set to 1 to enable. This bit is not banked between security states.		0
			0b	SecureFault exception is disabled	
			1b	SecureFault exception is enabled	

Table 1031. System handler control and state register - Offset: 24h...continued

Bit	Name	Access	Description and value	Reset value
18	USGFAULTENA	R/W	UsageFault enable Set to 1 to enable. This bit is banked between security states.	0
			0b UsageFault is disabled	
			1b UsageFault is enabled	
17	BUSFAULTENA	R/W	BusFault enable Set to 1 to enable. If AIRCR.BFHFNMINS is zero, this bit is RAZ/WI from non-secure state. This bit is not banked between security states.	0
			0b BusFault is disabled	
			1b BusFault is enabled	
16	MEMFAULTENA	R/W	MemManage enable Set to 1 to enable. This bit is banked between security states.	0
			0b MemManage exception is disabled	
			1b MemManage exception is enabled	
15	SVCALLPENDEd	R/W	SVCcall pending Reads as 1 if exception is pending. This bit is banked between security states.	0
			0b SVCcall exception is not pending	
			1b SVCcall exception is pending	
14	BUSFAULTPENDEd	R/W	BusFault exception pending Reads as 1 if exception is pending. If AIRCR.BFHFNMINS is zero, this bit is RAZ/WI from non-secure state. This bit is not banked between security states.	0
			0b BusFault exception is pending	
			1b BusFault exception is not pending	
13	MEMFAULTPENDEd	R/W	MemManage exception pending Reads as 1 if exception is pending. This bit is banked between security states.	0
			0b MemManage exception is not pending	
			1b MemManage exception is pending	
12	USGFAULTPENDEd	R/W	UsageFault exception pending Reads as 1 if exception is pending. This bit is banked between security states.	0
			0b UsageFault exception is not pending	
			1b UsageFault exception is pending	

Table 1031. System handler control and state register - Offset: 24h...continued

Bit	Name	Access	Description and value	Reset value
11	SYSTICKACT	R/W	SysTick exception active Reads as 1 if exception is active. This bit is banked between security states.	0
			0b SysTick exception is not active	
			1b SysTick exception is active	
10	PENDSVACT	R/W	PendSV exception active Reads as 1 if exception is active. This bit is banked between security states.	0
			0b PendSV exception is not active	
			1b PendSV exception is active	
9	reserved	-	reserved	-
8	MONITORACT	R/W	debug monitor active Reads as 1 if debug monitor is active. This bit is not banked between security states.	0
			0b debug monitor exception is not active	
			1b debug monitor exception is active	
7	SVCALLACT	R/W	SVCcall active Reads as 1 if SVC call is active. This bit is banked between security states.	0
			0b SVCcall exception is not active	
			1b SVCcall exception is active	
6	reserved	-	reserved	-
5	NMIACT	R/W	NMI exception active Reads as 1 if exception is active. This bit is not banked between security states.	0
			0b NMI exception is not active	
			1b NMI exception is active	
4	SECUREFAULTACT	R/W	SecureFault exception active Reads as 1 if exception is active. This bit is not banked between security states.	0
			0b SecureFault exception is not active	
			1b SecureFault exception is active	
3	USGFAULTACT	R/W	UsageFault exception active Reads as 1 if exception is active. This bit is banked between security states.	0
			0b UsageFault exception is not active	
			1b UsageFault exception is active	

Table 1031. System handler control and state register - Offset: 24h...continued

Bit	Name	Access	Description and value	Reset value
2	HARDFULTACT	R/W	HardFault exception active Reads as 1 if exception is active. This bit is banked between security states.	0
			0b HardFault exception is not active	
			1b HardFault exception is active	
1	BUSFAULTACT	R/W	BusFault exception active Reads as 1 if exception is active. If AIRCR.BFHFNMINS is zero, this bit is RAZ/WI from non-secure state. This bit is not banked between security states.	0
			0b BusFault exception is not active	
			1b BusFault exception is active	
0	MEMFAULTACT	R/W	MemManage exception active Reads as 1 if exception is active. This bit is banked between security states.	0
			0b MemManage exception is not active	
			1b MemManage exception is active	

45.6.2.2.4 Non-secure access control register (NSACR)

In an implementation with the security extension, the NSACR register defines the non-secure access permissions for the FPU and the coprocessors CP m, bit[m]. For m = 0 to 7.

Table 1032. Non-secure access control register - Offset: 8Ch

Bit	Name	Access	Description and value	Reset value
31:12	reserved	-	reserved	-
11	CP11	R/W	CP11 access Enables non-secure access to the floating-point extension. Programming with a different value other than that used for CP10 is UNPREDICTABLE. If the floating-point extension is not implemented, this bit is RAZ/WI.	0
10	CP10	R/W	CP10 access Enables non-secure access to the floating-point extension. If the floating-point extension is not implemented, this bit is RAZ/WI.	0
			0b Non-secure accesses to the floating-point extension generates a NOCP UsageFault.	
			1b Non-secure access to the floating-point extension is permitted.	
9:8	reserved	-	reserved	-

Table 1032. Non-secure access control register - Offset: 8Ch...continued

Bit	Name	Access	Description and value	Reset value
7	CP7	R/W	CP7 access Enables non-secure access to coprocessor CP7. If the CP7 is not implemented, this bit is RAZ/WI.	0
			0b Non-secure accesses to this coprocessor generate a NOCP UsageFault.	
			1b Non-secure accesses to this coprocessor is permitted.	
6	CP6	R/W	CP6 access Enables non-secure access to coprocessor CP6. If the CP6 is not implemented, this bit is RAZ/WI.	0
			0b Non-secure accesses to this coprocessor generate a NOCP UsageFault.	
			1b Non-secure accesses to this coprocessor is permitted.	
5	CP5	R/W	CP5 access Enables non-secure access to coprocessor CP5. If the CP5 is not implemented, this bit is RAZ/WI.	0
			0b Non-secure accesses to this coprocessor generate a NOCP UsageFault.	
			1b Non-secure accesses to this coprocessor is permitted.	
4	CP4	R/W	CP4 access Enables non-secure access to coprocessor CP4. If the CP4 is not implemented, this bit is RAZ/WI.	0
			0b Non-secure accesses to this coprocessor generate a NOCP UsageFault.	
			1b Non-secure accesses to this coprocessor is permitted.	
3	CP3	R/W	CP3 access Enables non-secure access to coprocessor CP3. If the CP3 is not implemented, this bit is RAZ/WI.	0
			0b Non-secure accesses to this coprocessor generate a NOCP UsageFault.	
			1b Non-secure accesses to this coprocessor is permitted.	
2	CP2	R/W	CP2 access Enables non-secure access to coprocessor CP2. If the CP2 is not implemented, this bit is RAZ/WI.	0
			0b Non-secure accesses to this coprocessor generate a NOCP UsageFault.	
			1b Non-secure accesses to this coprocessor is permitted.	
1	CP1	R/W	CP1 access Enables non-secure access to coprocessor CP1. If the CP1 is not implemented, this bit is RAZ/WI.	0
			0b Non-secure accesses to this coprocessor generate a NOCP UsageFault.	
			1b Non-secure accesses to this coprocessor is permitted.	

Table 1032. Non-secure access control register - Offset: 8Ch...continued

Bit	Name	Access	Description and value	Reset value
0	CP0	R/W	CP1 access Enables non-secure access to coprocessor CP1. If the CP1 is not implemented, this bit is RAZ/WI.	0
			0b Non-secure accesses to this coprocessor generate a NOCP UsageFault.	
			1b Non-secure accesses to this coprocessor is permitted.	

45.6.3 Security attribution unit (SAU)

45.6.3.1 SAU memory map

The SAU determines the security of an address.

The SAU registers are located at base address: E000_ED00h.

Table 1033. Security attribution unit registers

Name	Access	Offset	Description	Width (bits)	Reset value	Section
CTRL	R/W	D0h	security attribution unit control register	32	0000_0000h	Section 45.6.3.2.1
TYPE	R/W	D4h	security attribution unit type register	32	0000_0000h	Section 45.6.3.2.2
RNR	R/W	D8h	security attribution unit region number register	32		Section 45.6.3.2.3
RBAR	R/W	DCh	security attribution unit region base address register	32		Section 45.6.3.2.4
RLAR	R/W	E0h	security attribution unit region limit address register	32		Section 45.6.3.2.5
SFSR	R/W	E4h	secure fault status register	32	0000_0000h	Section 45.6.3.2.6
SFAR	R/W	E8h	secure fault address register	32		Section 45.6.3.2.7

45.6.3.2 Register descriptions

45.6.3.2.1 Security attribution unit control register (CTRL)

The CTRL register allows enabling of the security attribution unit.

In an implementation with the security extension, this register is:

- RAZ/WI when accessed as non-secure
- Not banked between security states

Table 1034. Security attribution unit control register - Offset: D0h

Bit	Name	Access	Description and value	Reset value
31:2	reserved	-	reserved	-
1	ALLNS	R/W	All non-secure When the ENABLE bit is 0, this bit controls if the memory is marked as non-secure or secure. This bit has no effect when ENABLE bit is 1. Setting the ALLNS bit to 1 allows the security attribution of all addresses to be set by the IDAU in the system.	0
			0b memory marked as secure and is not non-secure callable	
			1b memory is marked as non-secure	
0	ENABLE	R/W	enables the SAU This bit is RAZ/WI when the security extension is implemented without an SAU region.	0
			0b SAU is disabled	
			1b SAU is enabled	

45.6.3.2.2 Security attribution unit type register (TYPE)

The TYPE register indicates the number of regions implemented by the security attribution unit.

In an implementation with the security extension, this register is:

- RAZ/WI when accessed as non-secure
- Not banked between security states

Table 1035. Security attribution unit type register - Offset: D4h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved	-
7:0	SREGION	R/W	SAU regions The number of implemented SAU regions.	0

45.6.3.2.3 Security attribution unit region number register (RNR)

The RNR register selects the region currently accessed by the RBAR and RLAR registers.

In an implementation with the security extension, this register is:

- RAZ/WI when accessed as non-secure
- Not banked between security states

Table 1036. Security attribution unit region number register - Offset: D8h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved	-
7:0	REGION	R/W	region number Indicates the SAU region accessed by the RBAR and RLAR registers. If no SAU regions are implemented, this field is reserved. Writing a value corresponding to an unimplemented region is CONSTRAINED UNPREDICTABLE. This field resets to an UNKNOWN value on a warm reset.	0

45.6.3.2.4 Security attribution unit region base address register (RBAR)

The RBAR register provides indirect read and write access to the base address of the currently selected SAU region.

In an implementation with the security extension, this register is:

- RAZ/WI when accessed as non-secure
- Not banked between security states

Table 1037. Security attribution unit region base address register - Offset: DCh

Bit	Name	Access	Description and value	Reset value
31:5	BADDR	R/W	base address Holds bits [31:5] of the base address for the selected SAU region. Bits [4:0] of the base address are defined as 0x00.	0
4:0	reserved	-	reserved	-

45.6.3.2.5 Security attribution unit region limit address register (RLAR)

The RLAR register provides indirect read and write access to the limit address of the currently selected SAU region.

In an implementation with the security extension, this register is:

- RAZ/WI when accessed as non-secure
- Not banked between security states

Table 1038. Security attribution unit region limit address register - Offset: E0h

Bit	Name	Access	Description and value	Reset value
31:5	LADDR	R/W	limit address Holds bits [31:5] of the limit address for the selected SAU region. Bits [4:0] of the limit address are defined as 0x1F.	0
4:2	reserved	-	reserved	-
1	NSC	R/W	non-secure callable Controls if the non-secure state is permitted to execute an SG instruction from this region.	0
			0b region is not non-secure callable	
			1b region is non-secure callable	
0	ENABLE	R/W	SAU region enable	0
			0b SAU region is enabled	
			1b SAU region is disabled	

45.6.3.2.6 Secure fault status register (SFSR)

The SFSR provides information about security related faults.

In an implementation with the security extension, this register is:

- RAZ/WI when accessed as non-secure
- Not banked between security states

Table 1039. Secure fault status register - Offset: E4h

Bit	Name	Access	Description and value	Reset value
31:8	reserved	-	reserved	-
7	LSERR	R/W	lazy state error flag Sticky flag indicating that an error occurred during lazy state activation or deactivation.	0
			0b no error has occurred	
			1b error has occurred	

Table 1039. Secure fault status register - Offset: E4h...continued

Bit	Name	Access	Description and value	Reset value
6	SFARVALID	R/W	secure fault address valid This bit is set when the SFAR register contains a valid value. As with similar fields, such as BFSR.BFARVALID and MMFSR.MMARVALID, this bit can be cleared by other exceptions, such as BusFault.	0
			0b SFAR content not valid	
			1b SFAR content valid	
5	LSPERR	R/W	lazy state preservation error flag Sticky flag indicating that an SAU or IDAU violation occurred during the lazy preservation of floating-point state.	0
			0b no error has occurred	
			1b error has occurred	
4	INVTRAN	R/W	invalid transition flag Sticky flag indicating that an exception was raised due to a branch that was not flagged as being domain crossing causing a transition from secure to non-secure memory.	0
			0b no error has occurred	
			1b error has occurred	
3	AUVIOL	R/W	attribution unit violation flag Sticky flag indicating that an attempt was made to access parts of the address space that are marked as secure with NS-Req for the transaction set to non-secure. This bit is not set if the violation occurred during: lazy state preservation (see LSPERR. - Vector fetches).	0
			0b no error has occurred	
			1b error has occurred	
2	INVER	R/W	invalid exception return flag Can be caused by EXC_RETURN.DCRS being set to 0 when returning from an exception in the non-secure state. Or by EXC_RETURN.ES being set to 1 when returning from an exception in the non-secure state.	0
			0b no error has occurred	
			1b error has occurred	
1	INVIS	R/W	invalid integrity signature flag This bit is set if the integrity signature in an exception stack frame is found to be invalid during the unstacking operation.	0
			0b no error has occurred	
			1b error has occurred	

Table 1039. Secure fault status register - Offset: E4h...continued

Bit	Name	Access	Description and value	Reset value
0	INVEP	R/W	invalid entry point This bit is set if a function call from the non-secure state or exception targets a non-SG instruction in the secure state. This bit is also set if the target address is an SG instruction. However, there is no matching SAU/IDAU region with the NSC flag set.	0
			0b no error has occurred	
			1b error has occurred	

45.6.3.2.7 Secure fault address register (SFAR)

The SFAR shows the address of the memory location that caused a security violation.

In an implementation with the security extension, this register is:

- RAZ/WI when accessed as non-secure
- Not banked between security states

Table 1040. Secure fault address register - Offset: E8h

Bit	Name	Access	Description and value	Reset value
31:0	ADDRESS	R/W	When the SFARVALID bit of the SFSR is set to 1, this field holds the address of an access that caused an SAU violation.	0

46 References

- [1] **Cortex-M33 user manual** — Cortex-M33 DGUG ARM Cortex-M33 Devices Generic User Guide
- [2] **AN11538 application note** — SCTimer/PWM cookbook; 2016, NXP Semiconductors
<https://www.nxp.com/docs/en/application-note/AN11538.zip>
- [3] **UM10204 user manual** — I²C-bus specification and user manual
- [4] **KW45 BLE Host stack documentation** —
 - 1. Bluetooth Low Energy Application Developers Guide
 - 2. Bluetooth Low Energy Host Stack API Reference Manual
 - 3. Bluetooth Low Energy Quick Start GuideSee MCX W23 SDK documentation
- [5] **Register description** — [Register description](#) of MCXW23RM
- [6] **AN12283 application note** — LPC55Sxx Secure Boot; 2020, NXP Semiconductors
- [7] **MCX W23 data sheet** — Ultra-low power, small footprint Bluetooth Low Energy solution with integrated flash and security for IoT; 2025, NXP Semiconductors
- [8] **MBOOTELFTOSBUG user guide** — elftosb User's Guide; 2020, NXP Semiconductors
- [9] **AN14658 application note** — Hardware design considerations for MCX W23; 2025, NXP Semiconductors
- [10] **SMBus specification** — System Management Bus (SMBus) Specification; Version 3.2; 12 Jan 2022

47 Abbreviations

Table 1041. Abbreviations

Acronym	Description
ADC	analog-to-digital converter
AHB	advanced high-performance bus
AMBA	advanced microcontroller bus architecture
APB	advanced peripheral bus
API	application programming interface
BOD	brownout detection
BSDL	boundary-scan description language
CC	credential constraint
CRC	cyclic redundancy check
DC	debug credential
DCC	debug communication channel
DCFG	device configuration
DCK	debug credential key
DMA	direct memory access
EMC	external memory controller
FIFO	first-in-first-out
FRO	free-running oscillator
GPIO	general-purpose input/output
IAP	in-application programming
ISP	in-system programming
ISR	interrupt service routine
JTAG	joint test action group
LIN	local interconnect network
NVIC	nested vectored interrupt controller
PDM	pulse density modulation
PLL	phase-locked loop
POR	power-on reset
PWM	pulse width modulator
RAM	random access memory
RoT	root of trust
RoTpub	RoT public key
RoTID	RoT identifier
RoTMETA	RoT metadata
SDIO	secure digital input output

Table 1041. Abbreviations...continued

Acronym	Description
SPI	serial peripheral interface
SRAM	static random access memory
SWD	serial-wire debug
TAP	test access port
TBSA-M	Arm trusted base system architecture for ARMv8-M
USART	universal synchronous/asynchronous receiver/transmitter
VAD	voice activity detect
VU	vendor usage

48 Revision history

Table 1042. Revision history

Document ID	Release date	Description
MCXW23 v.3.0	17 Oct 2025	Added MCX W23xA
UM12312 v.2.1	08 Aug 2025	- Updated flas memory size for MCX235x from 640 kB to 512 kB in the document. - Updated Non-secure end address and Secure end address, and flash size for AHB port '0' in Table 4. - Removed internal content.
UM12312 v.2.0	20 June 2025	Initial version

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 <https://www.nxp.com/profile/terms>, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Tables

Tab. 1.	Ordering information	8	Tab. 47.	Framing packet format	82
Tab. 2.	Ordering options	8	Tab. 48.	Special framing packet format	83
Tab. 3.	TrustZone and system general mapping	10	Tab. 49.	Packet type field	83
Tab. 4.	Memory map overview	11	Tab. 50.	GetProperty command packet format (example)	83
Tab. 5.	APB peripherals memory map	12	Tab. 51.	CRC16 algorithm	84
Tab. 6.	AHB peripheral memory map	13	Tab. 52.	Ping packet format	85
Tab. 7.	SRAM configuration	14	Tab. 53.	Ping response packet format	85
Tab. 8.	Connection of interrupt sources to the NVIC	15	Tab. 54.	Command packet format	86
Tab. 9.	SYSCON pin description	20	Tab. 55.	Command header format	86
Tab. 10.	Clocking diagram signal name descriptions	20	Tab. 56.	Command tags	87
Tab. 11.	Boot mode and ISP download modes based on ISP pin	31	Tab. 57.	Response tags	87
Tab. 12.	ISP download mode based on DEFAULT_ ISP_MODE bits (BOOT_CFG[6:4], word 0 in CMPA)	31	Tab. 58.	GenericResponse parameters	88
Tab. 13.	ISP pin assignments	32	Tab. 59.	GetPropertyResponse parameters	88
Tab. 14.	Image header for the MCX W23 device	33	Tab. 60.	ReadMemoryResponse parameters	89
Tab. 15.	Image header for the MCX W23 devices	36	Tab. 61.	KeyProvisionResponse parameters	89
Tab. 16.	MbedTLS modules that are part of ROM	36	Tab. 62.	Ack packet format	89
Tab. 17.	MbedTLS KSDK modules that are part of ROM	39	Tab. 63.	Nak packet format	89
Tab. 18.	CMPA secure boot configuration overview	44	Tab. 64.	AckAbort packet format	90
Tab. 19.	SECURE_BOOT_CFG word bit field definitions	44	Tab. 65.	Parameters for GetProperty Command	90
Tab. 20.	PRINCE configuration	46	Tab. 66.	GetProperty command packet format (example)	91
Tab. 21.	PRINCE subregion enable bits	47	Tab. 67.	GetProperty response packet format (example)	92
Tab. 22.	RKTH layout in CMPA	48	Tab. 68.	GetProperty response status codes	92
Tab. 23.	CFPA page layout	49	Tab. 69.	Parameters for SetProperty command	92
Tab. 24.	RKTH_REVOKE table bit field description	50	Tab. 70.	SetProperty command packet format (example)	93
Tab. 25.	PUF key code storage area structure	51	Tab. 71.	SetProperty response status codes	94
Tab. 26.	PUF key code format	52	Tab. 72.	Parameter for FlashEraseAll command	94
Tab. 27.	MCX W23 image type (word at offset 0x24)	55	Tab. 73.	FlashEraseAll command packet format (example)	95
Tab. 28.	Standard Arm Cortex-M33 NVIC vector table	56	Tab. 74.	FlashEraseAll response status codes	95
Tab. 29.	Certificate block header fields	57	Tab. 75.	Parameter for FlashEraseRegion command ...	95
Tab. 30.	TrustZone image type	61	Tab. 76.	FlashEraseRegion response status codes	96
Tab. 31.	Miscellaneous TZM settings	64	Tab. 77.	Parameter for read memory command	97
Tab. 32.	TZM Preset value	65	Tab. 78.	ReadMemory command packet format (example)	98
Tab. 33.	Allowed TrustZone image types	66	Tab. 79.	ReadMemory response status codes	98
Tab. 34.	Structure for configure-memory command	69	Tab. 80.	Parameters for WriteMemory command	99
Tab. 35.	PRINCE configuration register for configure-memory command	69	Tab. 81.	WriteMemory command packet format (example)	100
Tab. 36.	PRINCE region info register for configure- memory command	69	Tab. 82.	WriteMemory response status codes	100
Tab. 37.	Secure ROM API summary	71	Tab. 83.	Parameters for FillMemory command	101
Tab. 38.	kb_authenticate status codes	72	Tab. 84.	FillMemory command packet format (example)	102
Tab. 39.	Parameters	73	Tab. 85.	FillMemory response status codes	102
Tab. 40.	bus_crypto_engine_set_encrypt_for_ address_range status codes	74	Tab. 86.	Parameters for execute command	102
Tab. 41.	Parameters	74	Tab. 87.	Execute response status codes	103
Tab. 42.	kb_init status codes	74	Tab. 88.	Reset command packet format (example)	103
Tab. 43.	Parameters	75	Tab. 89.	Parameters for ConfigureMemory command	104
Tab. 44.	kb_deinit status codes	75	Tab. 90.	ConfigureMemory response status codes	104
Tab. 45.	Parameters	76	Tab. 91.	Supported memory IDs	105
Tab. 46.	kb_execute status codes	76	Tab. 92.	Serial NOR flash configuration option block ..	105
			Tab. 93.	Parameters for Receive SB File command	106

Tab. 94.	ReceiveSBFile response status codes	106	Tab. 145.	POWER_ConfigureBOD API routines	170
Tab. 95.	Parameters for KeyProvision command	107	Tab. 146.	POWER_ClearBODInterrupt API routine	171
Tab. 96.	Key operation details	107	Tab. 147.	POWER_IsBODActive API routine	171
Tab. 97.	Key Type details	107	Tab. 148.	POWER_DCDC_GetSupplyMode API routine	171
Tab. 98.	KeyProvisioning command packet format (example)	108	Tab. 149.	POWER_DCDC_SetSupplyMode API routine	172
Tab. 99.	KeyProvisioning response packet format (Example)	109	Tab. 150.	POWER_DCDC_Disable and POWER_DCDC_Enable API routine	172
Tab. 100.	KeyProvisioning response status codes	109	Tab. 151.	POWER_DCDC_ConfigureBuckOutput API routine	172
Tab. 101.	Properties used by Get/SetProperty commands, sorted by values	109	Tab. 152.	POWER_DCDC_GetStartupDuration API routine	173
Tab. 102.	CurrentVersion property fields	111	Tab. 153.	POWER_XTAL32K_ConfigureCaps API routine	173
Tab. 103.	Peripheral bits	111	Tab. 154.	POWER_XTAL32K_ConfigureReferenceOutputCurrent API routine	173
Tab. 104.	Command bits	111	Tab. 155.	POWER_XTAL32K_ConfigureOscillatorAmplitude API routine	173
Tab. 105.	Fields of ExternalMemoryAttributes property	112	Tab. 156.	POWER_XTAL32K_EnableBypass API routine	174
Tab. 106.	Response word and error description	112	Tab. 157.	POWER_XTAL32K_SetStartupTime API routine	174
Tab. 107.	Command and response tags	114	Tab. 158.	POWER_XTAL32M_BLE_GateEnable API routine	174
Tab. 108.	Bootloader status error codes, sorted by value	114	Tab. 159.	POWER_XTAL32M_SetStartupTime API routine	174
Tab. 109.	Flash ROM API summary	126	Tab. 160.	POWER_EnterSleep API routine	175
Tab. 110.	Parameters	127	Tab. 161.	POWER_EnterDeepSleep API routine	175
Tab. 111.	Parameters	128	Tab. 162.	POWER_EnterPowerDown API routine	177
Tab. 112.	Parameters	128	Tab. 163.	POWER_EnterDeepPowerDown API routine	178
Tab. 113.	Parameters	129	Tab. 164.	Parameter wakeup_io_ctrl	179
Tab. 114.	Parameters	129	Tab. 165.	POWER_EnterPowerOff API routine	179
Tab. 115.	Parameters	130	Tab. 166.	GPIO pins available	185
Tab. 116.	Properties	130	Tab. 167.	INPUT MUX pin description	190
Tab. 117.	Parameters	131	Tab. 168.	Pin interrupt registers for edge-sensitive and level-sensitive pins	199
Tab. 118.	Parameters	131	Tab. 169.	Pin interrupt registers for edge-sensitive and level-sensitive pins	206
Tab. 119.	Parameters	131	Tab. 170.	DMA0 requests and trigger multiplexers	212
Tab. 120.	Parameters	132	Tab. 171.	DMA1 requests and trigger multiplexers	212
Tab. 121.	Parameters	132	Tab. 172.	DMA with the I2C	213
Tab. 122.	Parameters	133	Tab. 173.	DMA trigger sources	213
Tab. 123.	Parameters	133	Tab. 174.	Channel descriptor	214
Tab. 124.	Parameters	133	Tab. 175.	Reload descriptors	215
Tab. 125.	ffr_key_type_t	134	Tab. 176.	Channel descriptor for a single transfer	216
Tab. 126.	Parameters	134	Tab. 177.	Example descriptors for Ping-Pong operation: peripheral to buffer	216
Tab. 127.	Parameters	135	Tab. 178.	SCT0 pin description (inputs)	221
Tab. 128.	Parameters	135	Tab. 179.	SCT0 pin description (outputs)	222
Tab. 129.	arg interpretation	135	Tab. 180.	Suggested SCT input pin settings	222
Tab. 130.	Flash and PFR driver status codes	136	Tab. 181.	Suggested SCT output pin settings	223
Tab. 131.	Lifecycle state descriptions	140	Tab. 182.	Event conditions	228
Tab. 132.	Example of capacitance value and gm setting for a certain crystal specification	148	Tab. 183.	SCT configuration example	233
Tab. 133.	XTAL32K API calls	148	Tab. 184.	Timer/counter pin description	238
Tab. 134.	XTAL32M API calls	149	Tab. 185.	Micro-tick timer pin description	242
Tab. 135.	Power domain supply	151			
Tab. 136.	Power modes	154			
Tab. 137.	Peripheral reduced power modes	154			
Tab. 138.	Power API calls	162			
Tab. 139.	Low power API calls	164			
Tab. 140.	Parameter exclude_from_pd	164			
Tab. 141.	Parameter wakeup_interrupts	165			
Tab. 142.	POWER_PeripheralPowerOff and POWER_PeripheralPowerOn API routine	169			
Tab. 143.	POWER_DisableSRAM and POWER_EnableSRAM API routine	169			
Tab. 144.	POWER_SetBodLevel API routines	170			

Tab. 186. RTC subsecond counter register (SUBSEC, offset 0x10)	247	Tab. 238. PHY (1 octet)	383
Tab. 187. RTC pin description	248	Tab. 239. Transmit_Power_Level (1 octet)	383
Tab. 188. Flexcomm interface base addresses and functions	265	Tab. 240. Status (1 octet)	383
Tab. 189. Flexcomm interface pin description	266	Tab. 241. Current_TX_Power_Level (1 octet)	384
Tab. 190. I2C-bus pin description	267	Tab. 242. HCI set RSSI golden range command	384
Tab. 191. Code example	269	Tab. 243. Golden_Range_Upper_Limit (1 octet)	384
Tab. 192. Code example	270	Tab. 244. Golden_Range_Lower_Limit (1 octet)	384
Tab. 193. Code example	271	Tab. 245. Status (1 octet)	384
Tab. 194. Code example	272	Tab. 246. HCI select TX power mode command	384
Tab. 195. Settings for the 400 kHz clock rate	274	Tab. 247. TX_Power_Mode (1 octet)	385
Tab. 196. Automatic operation cases	277	Tab. 248. Status (1 octet)	385
Tab. 197. USART pin description	281	Tab. 249. HCI curb sleep state command	385
Tab. 198. Suggested USART pin settings	282	Tab. 250. Enable (1 octet)	385
Tab. 199. SPI pin description	290	Tab. 251. Status (1 octet)	386
Tab. 200. Suggested SPI pin settings	291	Tab. 252. HCI locally terminate ACL connection command	386
Tab. 201. SPI mode summary	293	Tab. 253. Connection_Handle (2 octets)	386
Tab. 202. SPIFI flash memory map	301	Tab. 254. Status (1 octet)	386
Tab. 203. SPIFI pin description	302	Tab. 255. hci_rgstr_ble_external_custom_cbk() function	386
Tab. 204. Time interval register (INTVAL[0:3], offset = 0x000 (INTVAL0) to 0x030 (INTVAL3))	307	Tab. 256. Callback function	387
Tab. 205. Security tier granularity for on-chip memories	324	Tab. 257. Register categories	391
Tab. 206. Number of clock cycles per operation	333	Tab. 258. CFPA memory map overview	392
Tab. 207. Coding of KEYSIZE	334	Tab. 259. CFPA header register - offset: 0h	393
Tab. 208. KC header field description	335	Tab. 260. CFPA version register - offset: 4h	393
Tab. 209. Key target interfaces per key index	337	Tab. 261. Secure firmware version register - offset: 8h	393
Tab. 210. Function parameters	338	Tab. 262. Non-secure firmware version register - offset: Ch	393
Tab. 211. Data access functions	338	Tab. 263. Image key revocation ID register - offset: 10h	394
Tab. 212. Performance comparison for elliptic curve calculations	348	Tab. 264. Individual root keys revocation control - offset: 18h	394
Tab. 213. Casper AHB operations	352	Tab. 265. Reserved for vendor use - offset: 1Ch	394
Tab. 214. Serial Wire Debug pin description	354	Tab. 266. Security policy for debug access control pin configuration - offset: 20h	396
Tab. 215. Resets	357	Tab. 267. Security policy for debug access control default configuration - offset: 24h	397
Tab. 216. Request register byte description	359	Tab. 268. Enable FA mode - offset: 28h	398
Tab. 217. DM-AP commands	360	Tab. 269. CMPA page programming ongoing - offset: 2Ch	398
Tab. 218. Response register byte description	361	Tab. 270. Customer-defined CFPA fields - offset: 80h + (a × 4h)	398
Tab. 219. DM-AP response codes	362	Tab. 271. SHA256 digest for CFPA - offset: 1E0h + (a × 4h)	398
Tab. 220. ACK_TOKEN register byte description	362	Tab. 272. FLASH_CMPA base address: 0010_4000h ..	399
Tab. 221. Glossary	376	Tab. 273. Boot configuration - offset: 0h	400
Tab. 222. HCI LE set custom event mask command	379	Tab. 274. SPI flash configuration - offset: 4h	400
Tab. 223. LE custom event mask parameter (1 octet)	379	Tab. 275. Security policy for debug access control pin configuration - offset: 8h	401
Tab. 224. LE custom event mask return value (1 octet)	379	Tab. 276. Security policy for debug access control default configuration - offset: Ch	402
Tab. 225. HCI release version command	380	Tab. 277. Reserved for vendor use - offset: 10h	402
Tab. 226. HCI set PHY TX power command	380	Tab. 278. Secure boot configuration - offset: 14h	403
Tab. 227. HCI set BD address command	380	Tab. 279. Prince configuration - offset: 1Ch	405
Tab. 228. BD Address parameter (6 octets)	380	Tab. 280. Bitmap control for memory context 0 - offset: 20h	406
Tab. 229. HCI periodic PHY calibration command	381	Tab. 281. Bitmap control for memory context 1 - offset: 24h	406
Tab. 230. HCI set deep sleep mode enable command	381		
Tab. 231. HCI get remaining time for next event command	381		
Tab. 232. HCI set event setup time value command	382		
Tab. 233. HCI select sleep clock source command	382		
Tab. 234. Sleep clock source parameter (1 octet)	382		
Tab. 235. Sleep clock frequency (2 octets)	382		
Tab. 236. HCI set connection TX power command	383		
Tab. 237. Connection_Handle (2 octets)	383		

Tab. 282. Bitmap control for memory context 2 - offset: 28h	407	Tab. 324. NXP device certificate part 3	419
Tab. 283. Bitmap control for memory context 3 - offset: 28h	407	Tab. 325. NXP attestation private key - offset: 68Ch + (a × 4h)	419
Tab. 284. Flash remap size - offset: 58h	407	Tab. 326. Offset	420
Tab. 285. Flash remap offset - offset: 5Ch	408	Tab. 327. NXP attestation certificate part 0	420
Tab. 286. Flash remap offset - offset: 60h + (a × 4h)	408	Tab. 328. Offset	420
Tab. 287. Customer-defined CMPA fields - offset: 80h + (a × 4h)	408	Tab. 329. NXP attestation certificate part 1	420
Tab. 288. SHA256 digest for CMPA - offset: 1E0h + (a × 4h)	408	Tab. 330. Offset	420
Tab. 289. OEM KEY STORE base address: 0010_5000h	409	Tab. 331. NXP attestation certificate part 2	421
Tab. 290. Valid OEM key store header - offset: 0h	409	Tab. 332. Offset	421
Tab. 291. Secure boot key code header 0 - offset: 10h	409	Tab. 333. NXP attestation certificate part 3	421
Tab. 292. Secure boot key code header 1 - offset: 14h	410	Tab. 334. SHA256 digest for CMPA - offset: 7E0h + (a × 4h)	421
Tab. 293. Secure boot key code body - offset: 18h + (a × 4h)	410	Tab. 335. FLASH_ROMPATCH base address: 10_8000h	422
Tab. 294. User key encryption code header 0 - offset: 48h	410	Tab. 336. ROM patch header - offset: 0h	422
Tab. 295. User key encryption code header 1 - offset: 4Ch	410	Tab. 337. ROM patch code - offset: 4h + (a × 4h)	422
Tab. 296. User key encryption code body - offset: 50h + (a × 4h)	411	Tab. 338. FLASH_NMPA base address: 10_8000h	423
Tab. 297. OEM KEY STORE base address: 0010_7800h	412	Tab. 339. True random number generator control - offset: B00h	426
Tab. 298. Valid NXP key store header - offset: 0h	414	Tab. 340. True random number generator poker range - offset: B04h	427
Tab. 299. PUF discharge time - offset: 4h	414	Tab. 341. True random number generator poker maximum limit - offset: B08h	427
Tab. 300. PUF activation code - offset: 8h + (a × 4h)	414	Tab. 342. True random number generator seed control - offset: B0Ch	427
Tab. 301. Prince region 0 key header 0 - offset: 4B0h ..	414	Tab. 343. True random number generator sparse bit limit - offset: B10h	427
Tab. 302. Prince region 0 key header 1 - offset: 4B4h ..	414	Tab. 344. True random number generator frequency count minimum limit - offset: B14h	427
Tab. 303. Prince region 0 key header 1 - offset: 4B8h + (a × 4h)	415	Tab. 345. True random number generator frequency count maximum limit - offset: B18h	428
Tab. 304. Prince region 1 key header 0 - offset: 4E8h ..	415	Tab. 346. True random number generator poker range statistical check monobit limit - offset: B1Ch	428
Tab. 305. Prince region 1 key header 1 - offset: 4ECh ..	415	Tab. 347. True random number generator poker range statistical check run length 1 limit - offset: B20h	428
Tab. 306. Prince region 1 key code body - offset: 4F0h + (a × 4h)	416	Tab. 348. True random number generator poker range statistical check run length 2 limit - offset: B24h	428
Tab. 307. Prince region 2 key header 0 - offset: 520h ..	416	Tab. 349. True random number generator poker range statistical check run length 3 limit - offset: B28h	429
Tab. 308. Prince region 2 key header 1 - offset: 524h ..	416	Tab. 350. True random number generator poker range statistical check run length 4 limit - offset: B2Ch	429
Tab. 309. Prince region 2 key code body - offset: 528h + (a × 4h)	416	Tab. 351. True random number generator poker range statistical check run length 5 limit - offset: B30h	429
Tab. 310. Prince region 3 key header 0 - offset: 558h ..	416	Tab. 352. True random number generator poker range statistical check run length 6+ limit - offset: B34h	431
Tab. 311. Prince region 3 key header 1 - offset: 55Ch ..	417	Tab. 353. Device name and type - offset: B60h	431
Tab. 312. Prince region 3 key code body - offset: 560h + (a × 4h)	417	Tab. 354. Offset	432
Tab. 313. UDS key header 0 - offset: 590h	417	Tab. 355. Unique device identifier - offset: B70h (a × 4h)	432
Tab. 314. UDS key header 1 - offset: 594h	417	Tab. 356. Peripheral enable configuration - Offset: B80h	432
Tab. 315. UDS key body - offset: 598h + (a × 4h)	418		
Tab. 316. NXP device private key - offset: 60Ch + (a × 4h)	418		
Tab. 317. Offset	418		
Tab. 318. NXP device certificate part 0	418		
Tab. 319. Offset	418		
Tab. 320. NXP device certificate part 1	419		
Tab. 321. Offset	419		
Tab. 322. NXP device certificate part 2	419		
Tab. 323. Offset	419		

Tab. 357.	RAM size configuration - Offset: B84h	432	Tab. 398.	Buffering of write accesses on synchronous system configuration APB interface - Offset: 20h	454
Tab. 358.	Flash size configuration - Offset: B88h	432	Tab. 399.	Buffering of write accesses on synchronous system configuration APB interface - Offset: 24h	456
Tab. 359.	Disable ROM hiding - Offset: B8Ch	433	Tab. 400.	System tick calibration for secure part of CPU0 - Offset: 38h	458
Tab. 360.	Ring oscillator 0 control - Offset: B90h	433	Tab. 401.	System tick calibration for non-secure part of CPU0 - Offset: 3Ch	458
Tab. 361.	Ring oscillator 1 control - Offset: B94h	433	Tab. 402.	NMI source select - Offset: 48h	458
Tab. 362.	Ring oscillator 2 control - Offset: B98h	433	Tab. 403.	Peripheral reset control 0 - Offset: 100h	459
Tab. 363.	PUF SRAM control configuration - Offset: B9Ch	433	Tab. 404.	Peripheral reset control 1 - Offset: 104h	461
Tab. 364.	Brownout detector - Offset: BA0h	434	Tab. 405.	Peripheral reset control 2 - Offset: 108h	462
Tab. 365.	LDO_MEM - Offset: BA4h	435	Tab. 406.	Offset	465
Tab. 366.	LDO_DIG - Offset: BA8h	436	Tab. 407.	Peripheral reset control set register	465
Tab. 367.	DCDC high power profile - Offset: BC8h	438	Tab. 408.	Offset	465
Tab. 368.	SHA256 digest for CMPA - offset: BE0h + (a × 4h)	439	Tab. 409.	Peripheral reset control clear register	465
Tab. 369.	General FMU parameters - offset: 1600h)	439	Tab. 410.	Peripheral reset control clear register	465
Tab. 370.	General FMU parameters - offset: 1604h)	439	Tab. 411.	AHB clock control 0 - Offset: 200h	466
Tab. 371.	General FMU parameters - offset: 1608h)	439	Tab. 412.	AHB clock control 1 - Offset: 204h	467
Tab. 372.	General FMU parameters - offset: 160Ch)	439	Tab. 413.	AHB clock control 2 - Offset: 208h	469
Tab. 373.	SMW timer options, SMW setting options - offset: 1610h)	440	Tab. 414.	Offset	471
Tab. 374.	SMW timer options, SMW setting options - offset: 1614h)	440	Tab. 415.	Peripheral reset control set register	471
Tab. 375.	SMW timer options, SMW setting options - offset: 1618h)	440	Tab. 416.	Offset	471
Tab. 376.	SMW timer options, SMW setting options - offset: 161Ch)	440	Tab. 417.	Peripheral reset control set register	471
Tab. 377.	SMW setting options continued, SMW smart program/erase options - offset: 1620h)	440	Tab. 418.	System tick timer source select - Offset: 260h	471
Tab. 378.	SMW setting options continued, SMW smart program/erase options - offset: 1624h)	441	Tab. 419.	Trace clock source select - Offset: 268h	472
Tab. 379.	SMW setting options continued, SMW smart program/erase options - offset: 1628h)	441	Tab. 420.	CTimer 0 - 4 clock source select - Offset: 26Ch + (a × 4)	472
Tab. 380.	SMW setting options continued, SMW smart program/erase options - offset: 162Ch)	441	Tab. 421.	Main clock A source select - Offset: 280h)	473
Tab. 381.	SMW smart erase options continued - offset: 1630h)	441	Tab. 422.	Main clock B source select - Offset: 284h)	473
Tab. 382.	Flash array size - offset: 1640h)	441	Tab. 423.	CLKOUT clock source select - Offset: 288h)	473
Tab. 383.	Flash TME control - offset: 1650h)	442	Tab. 424.	Flexcomm interface 0 clock source select for fractional rate divider - Offset: 2B0h)	474
Tab. 384.	SoC trim settings enable - offset: 17E0h)	442	Tab. 425.	Flexcomm interface 1 clock source select for fractional rate divider - Offset: 2B4h)	474
Tab. 385.	SoC trim ECC decoder enable - offset: 17F0h)	442	Tab. 426.	Flexcomm interface 2 clock source select for fractional rate divider - Offset: 2B8h)	475
Tab. 386.	SOCTRIM0 FRO32K - offset: 1800h)	442	Tab. 427.	QuadSPI flash interface clock source select - Offset: 2D0h)	475
Tab. 387.	SOCTRIM0 flash speed - offset: 1808h)	443	Tab. 428.	SCTimer/PWM clock source select - Offset: 2F0h)	476
Tab. 388.	SOCTRIM1 device ID - offset: 1810h)	443	Tab. 429.	System tick timer divider for CPU0 - Offset: 300h)	476
Tab. 389.	SOCTRIM1 device revision - offset: 1814h)	444	Tab. 430.	TRACE clock divider - Offset: 308h)	477
Tab. 390.	SOCTRIM2 FRO192M - offset: 1820h)	444	Tab. 431.	Fractional rate divider for flexcomm 0 - Offset: 320h)	477
Tab. 391.	SOCTRIM3 band gap - offset: 1830h)	444	Tab. 432.	Fractional rate divider for flexcomm 1 - Offset: 324h)	477
Tab. 392.	RAM configuration	446	Tab. 433.	Fractional rate divider for flexcomm 2 - Offset: 324h)	478
Tab. 393.	External flash memory map	447	Tab. 434.	System clock divider - Offset: 380h)	478
Tab. 394.	APB peripherals memory map	448	Tab. 435.	CLKOUT clock divider - Offset: 384h)	478
Tab. 395.	SYSCON - base address: 4000_0000h	449	Tab. 436.	FRO_HF (32 MHz) clock divider - Offset: 388h)	479
Tab. 396.	Memory remap control register - Offset: 0h	453	Tab. 437.	WDT clock divider - Offset: 38Ch)	479
Tab. 397.	AHB matrix priority control register - Offset: 10h	453			

Tab. 438.	SCT/PWM clock divider - Offset: 3B4h)	479	Tab. 477.	Security code to allow test (design for testability) access - Offset: FB0h	504
Tab. 439.	QuadSPI flash interface clock divider - Offset: 3D0h)	480	Tab. 478.	SWD access control register - Offset: FB4h ..	504
Tab. 440.	Control clock configuration registers access - Offset: 3FCh)	480	Tab. 479.	Block PUF generated keys - Offset: FBCh	504
Tab. 441.	ROM access configuration register - Offset: 404h)	481	Tab. 480.	Debug authentication BEACON register - Offset: FC0h	504
Tab. 442.	EZH interrupt hijack - Offset: 414h)	481	Tab. 481.	Flash size configuration - Offset: FE0h	505
Tab. 443.	Flash remap size - Offset: 440h)	481	Tab. 482.	Configuration lockout register - Offset: FE4h	505
Tab. 444.	Flash remap size duplicate - Offset: 444h)	481	Tab. 483.	RAM size configuration - Offset: FE8h	506
Tab. 445.	Flash remap offset - Offset: 448h)	481	Tab. 484.	Peripheral enable configuration - Offset: FECh	507
Tab. 446.	Flash remap offset duplicate - Offset: 44Ch)	482	Tab. 485.	Device ID - Offset: FF8h	508
Tab. 447.	Flash banks control - Offset: 450h)	482	Tab. 486.	Device ID - Offset: FFCh	510
Tab. 448.	Flash remap lock registers - Offset: 45Ch)	484	Tab. 487.	IOCON - base address: 4000_1000h	511
Tab. 449.	CASPER control register - Offset: 470h)	484	Tab. 488.	Digital I/O control for port 0 pins PIO0_b - Offset: 0h + (b × 4h)	511
Tab. 450.	Boot lockout address offset - Offset: 480h)	484	Tab. 489.	Offset	512
Tab. 451.	Boot lockout address offset duplicate - Offset: 484h)	484	Tab. 490.	Digital I/O control for port 0 pins PIO0_13 and PIO0_14	512
Tab. 452.	Boot lockout register - Offset: 488h)	485	Tab. 491.	GINT - base address: 4000_1000h	514
Tab. 453.	Start logic wake-up enable register - Offset: 680h	485	Tab. 492.	GPIO grouped interrupt control register - Offset: 0h	514
Tab. 454.	Start logic wake-up enable register - Offset: 684h)	488	Tab. 493.	GPIO grouped interrupt polarity register - Offset: 20h	515
Tab. 455.	Start logic wake-up enable set 0 register - Offset: 6A0h)	490	Tab. 494.	GPIO grouped interrupt port 0 enable register - Offset: 40h	515
Tab. 456.	Start logic wake-up enable set 1 register - Offset: 6A4h)	491	Tab. 495.	PINT - base address: 4000_1000h	516
Tab. 457.	Start logic wake-up enable clear 0 register - Offset: 6C0h)	492	Tab. 496.	Pin interrupt mode register - Offset: 0h	517
Tab. 458.	Start logic wake-up enable clear 1 register - Offset: 6C4h)	494	Tab. 497.	Pin interrupt level or rising-edge interrupt enable register - Offset: 4h	517
Tab. 459.	Functional retention control register - Offset: 704h)	495	Tab. 498.	Pin interrupt level or rising-edge interrupt set register - Offset: 8h	517
Tab. 460.	Power-down control override signal - Offset: 70Ch)	495	Tab. 499.	Pin interrupt level (rising edge interrupt) clear register - Offset: Ch	518
Tab. 461.	Main clock safety cycles register - Offset: 710h)	495	Tab. 500.	Pin interrupt active level or falling edge interrupt enable register - Offset: 10h	518
Tab. 462.	Hardware sleep control - Offset: 780h)	496	Tab. 501.	Pin interrupt active level or falling-edge interrupt clear register - Offset: 18h	518
Tab. 463.	CPU status - Offset: 80Ch	497	Tab. 502.	Pin interrupt active level or falling-edge interrupt clear register - Offset: 18h	519
Tab. 464.	CPU status - Offset: 900h + (n × 4h)	497	Tab. 503.	Pin interrupt rising-edge register - Offset: 1Ch	519
Tab. 465.	Boot seed register - Offset: 920h + (n × 4h) ..	497	Tab. 504.	Pin interrupt falling-edge register - Offset: 20h	519
Tab. 466.	HMAC register - Offset: 940h + (n × 4h)	497	Tab. 505.	Pin interrupt status register - Offset: 24h	520
Tab. 467.	Boot seed security register - Offset: 960h	498	Tab. 506.	Pattern match interrupt control register - Offset: 28h	520
Tab. 468.	Code watchdog status register - Offset: 9D0h	498	Tab. 507.	Pattern match interrupt bit-slice source register - Offset: 2Ch	521
Tab. 469.	Clock control register - Offset: A18h	498	Tab. 508.	Pattern match interrupt bit slice configuration register - Offset: 30h	527
Tab. 470.	Control automatic clock gating - Offset: E04h	500	Tab. 509.	INPUTMUX - base address: 4000_6000h	530
Tab. 471.	GPIO synchronization control - Offset: E08h	502	Tab. 510.	Offset	532
Tab. 472.	Invert main clock - Offset: E20h	502	Tab. 511.	Input mux register for SCT0 input - Offset: 30h	532
Tab. 473.	HASH AES key restricted use register - Offset: F88h	502	Tab. 512.	Offset	533
Tab. 474.	Debug lock enable - Offset: FA0h	502	Tab. 513.	Capture select registers for TIMER0 inputs ...	533
Tab. 475.	Cortex debug features control - Offset: FA4h	503	Tab. 514.	Offset	534
Tab. 476.	Cortex debug features control duplicate - Offset: FA8h	503			

Tab. 515.	Capture select registers for TIMER1 inputs ...	534	Tab. 557.	Capture register	556
Tab. 516.	Offset	536	Tab. 558.	External match register - Offset: 3Ch	556
Tab. 517.	Capture select registers for TIMER2 inputs ...	536	Tab. 559.	Count control register - Offset: 70h	559
Tab. 518.	Input mux register for EZH arch B inputs - Offset: A0h + (a × 4h)	538	Tab. 560.	PWM control register - Offset: 74h	560
Tab. 519.	Pin interrupt select register - Offset: C0h + (a × 4h)	539	Tab. 561.	Offset	561
Tab. 520.	Trigger select register for DMA0 channel - Offset: E0h + (a × 4h)	540	Tab. 562.	Match shadow register	561
Tab. 521.	Offset	541	Tab. 563.	Windowed watchdog timer registers	562
Tab. 522.	DMA0 output trigger selection to become DMA0 trigger	541	Tab. 564.	Watchdog mode register - Offset: 0h	563
Tab. 523.	Selection for frequency measurement reference	clock	Tab. 565.	Watchdog timer constant register - Offset: 4h	564
Tab. 524.	Selection for frequency measurement target clock - Offset: 184h	542	Tab. 566.	Watchdog feed sequence register - Offset: 8h	564
Tab. 525.	Offset	542	Tab. 567.	Watchdog timer value register - Offset: Ch	564
Tab. 526.	Capture select registers for TIMER3 inputs ...	543	Tab. 568.	Watchdog warning interrupt compare value - Offset: 14h	564
Tab. 527.	Offset	543	Tab. 569.	Watchdog window compare value - Offset: 18h	565
Tab. 528.	Capture select registers for TIMER4 inputs ...	544	Tab. 570.	Multirate timer registers	566
Tab. 529.	Offset	544	Tab. 571.	Offset	567
Tab. 530.	Pin interrupt secure select register	545	Tab. 572.	MRT time interval value register	567
Tab. 531.	Trigger select register for DMA0 channel - Offset: E0h + (a × 4h)	545	Tab. 573.	Offset	568
Tab. 532.	Offset	545	Tab. 574.	MRT timer register	568
Tab. 533.	DMA1 output trigger selection to become DMA1 trigger	546	Tab. 575.	Offset	568
Tab. 534.	Enable DMA0 requests - Offset: 740h	546	Tab. 576.	MRT control register	569
Tab. 535.	Set one or several bits in DMA0_REQ_ ENA register - Offset: 748h	547	Tab. 577.	Offset	569
Tab. 536.	Clear one or several bits in DMA0_REQ_ ENA register - Offset: 750h	547	Tab. 578.	MRT control register	569
Tab. 537.	Enable DMA1 requests - Offset: 760h	547	Tab. 579.	Module configuration register - Offset: F0h ...	570
Tab. 538.	Set one or several bits in DMA1_REQ_ ENA register - Offset: 768h	548	Tab. 580.	Idle channel register - Offset: F4h	571
Tab. 539.	Clear one or several bits in DMA1_REQ_ ENA register - Offset: 770h	548	Tab. 581.	Global interrupt flag register - Offset: F8h	571
Tab. 540.	Enable DMA0 triggers - Offset: 780h	548	Tab. 582.	Micro-tick timer registers	572
Tab. 541.	Set one or several bits in DMA0_ITRIG_ ENA register - Offset: 788h	548	Tab. 583.	Control register - Offset: 0h	572
Tab. 542.	Clear one or several bits in DMA0_ITRIG_ ENA register - Offset: 790h	548	Tab. 584.	Status register - Offset: 4h	573
Tab. 543.	Enable DMA1 triggers - Offset: 7A0h	549	Tab. 585.	Capture configuration register - Offset: 8h	573
Tab. 544.	Set one or several bits in DMA1_ITRIG_ ENA register - Offset: 7A8h	549	Tab. 586.	Capture clear register - Offset: Ch	574
Tab. 545.	Clear one or several bits in DMA1_ITRIG_ ENA register - Offset: 7B0h	549	Tab. 587.	Offset	574
Tab. 546.	CTIMER	550	Tab. 588.	Capture register	575
Tab. 547.	Interrupt register - Offset: 0h	551	Tab. 589.	Analog control registers	576
Tab. 548.	Timer control register - Offset: 4h	551	Tab. 590.	Flash status registers - Offset: 4h	577
Tab. 549.	Timer counter - Offset: 8h	552	Tab. 591.	Frequency measure function control register - Offset: Ch	577
Tab. 550.	Prescale register - Offset: Ch	552	Tab. 592.	192 MHz free running oscillator (FRO) clock control register - Offset: 10h	578
Tab. 551.	Prescale counter - Offset: 10h	552	Tab. 593.	192 MHz free running oscillator (FRO) status register - Offset: 14h	579
Tab. 552.	Match control register - Offset: 14h	552	Tab. 594.	Status of the analog components of the Bluetooth Low Energy radio - Offset: 28h	579
Tab. 553.	Offset	554	Tab. 595.	Bluetooth Low Energy - Offset: 2Ch	580
Tab. 554.	Match register	554	Tab. 596.	Brownout detectors (BoDs) and DCDC interrupt generation control register - Offset: 30h	580
Tab. 555.	Capture control register - Offset: 28h	554	Tab. 597.	BoDs and DCDC interrupt status register - Offset: 34h	581
Tab. 556.	Offset	556	Tab. 598.	First ring oscillator module control register - Offset: 40h	582
			Tab. 599.	Second ring oscillator module control register - Offset: 44h	583
			Tab. 600.	Third ring oscillator module control register - Offset: 48h	585
			Tab. 601.	Power management control registers	586

Tab. 602.	Power management control - Offset: 0h	589	Tab. 642.	System configuration registers	623
Tab. 603.	Power management controller FSM status - Offset: 4h	590	Tab. 643.	Update lockout control - Offset: 0h	624
Tab. 604.	Reset control - Offset: 8h	594	Tab. 644.	Auxiliary flash control - Offset: 100h	624
Tab. 605.	Reset cause register - Offset: Ch	595	Tab. 645.	Gray code LSB input register - Offset: 180h	625
Tab. 606.	DCDC control register - Offset: 10h	596	Tab. 646.	Gray code MSB input register - Offset: 184h	625
Tab. 607.	Bias current source control register - Offset: 18h	596	Tab. 647.	Binary code LSB output register - Offset: 188h	626
Tab. 608.	LDOMEM - Offset: 20h	597	Tab. 648.	Binary code MSB output register - Offset: 18Ch	626
Tab. 609.	LDODIG - Offset: 24h	597	Tab. 649.	Auxiliary random number generator control - Offset: 200h	626
Tab. 610.	BOD1 - Offset: 30h	597	Tab. 650.	Auxiliary random number generator status - Offset: 208h	626
Tab. 611.	BOD2 - Offset: 34h	598	Tab. 651.	Real-time clock registers	627
Tab. 612.	FRO192M control - Offset: 40h	599	Tab. 652.	RTC control register - Offset: 0h	627
Tab. 613.	FRO1M control - Offset: 44h	599	Tab. 653.	RTC match register - Offset: 4h	630
Tab. 614.	FRO32K control - Offset: 48h	599	Tab. 654.	RTC counter register - Offset: 8h	630
Tab. 615.	32 KHz crystal oscillator control register - Offset: 4Ch	600	Tab. 655.	High-resolution/wake-up timer control register - Offset: Ch	631
Tab. 616.	Bluetooth Low Energy xtal overrule - Offset: 54h	600	Tab. 656.	Subsecond counter register - Offset: 10h	631
Tab. 617.	Wake-up pin control - Offset: 64h	601	Tab. 657.	Subsecond counter register - Offset: 40h + (a × 4h)	631
Tab. 618.	Wake-up I/O source from deep power-down mode - Offset: 68h	602	Tab. 658.	Synchronous OS/event timer with wake-up timer registers	632
Tab. 619.	Status power - Offset: 70h	602	Tab. 659.	EVTIMER low register - Offset: 0h	632
Tab. 620.	FRO and xtal status register - Offset: 74h	602	Tab. 660.	EVTIMER high register - Offset: 4h	632
Tab. 621.	Write-once general purpose always on domain data storage - Offset: 80h	603	Tab. 661.	Capture low register - Offset: 8h	633
Tab. 622.	General purpose always on domain data storage - Offset: 84h	603	Tab. 662.	Capture high register - Offset: 8h	633
Tab. 623.	General purpose always on domain data storage - Offset: 88h	603	Tab. 663.	Match low register - Offset: Ch	633
Tab. 624.	RTC 1 KHz and 1 Hz clocks source control register - Offset: 98h	603	Tab. 664.	Match high register - Offset: 10h	633
Tab. 625.	OS timer control register - Offset: 9Ch	604	Tab. 665.	OS_EVENT TIMER control register - Offset: 1Ch	634
Tab. 626.	Timeout status events during IC power-up - Offset: A4h	605	Tab. 666.	Flash memory unit registers	635
Tab. 627.	Timeout - Offset: A8h	606	Tab. 667.	Flash status register - Offset: 0h	636
Tab. 628.	Power-down configuration during low-power mode - Offset: B0h	606	Tab. 668.	Flash configuration register - Offset: 4h	640
Tab. 629.	SRAM retention control - Offset: B4h	610	Tab. 669.	Flash control register - Offset: 8h	642
Tab. 630.	Power-down configuration during active mode - Offset: B8h	615	Tab. 670.	Flash test register - Offset: 8h	644
Tab. 631.	Power-down configuration during active mode set register - Offset: C0h	616	Tab. 671.	Flash common command object registers - Offset: 10h + (a × 4h)	645
Tab. 632.	Power-down configuration during active mode clear register - Offset: C8h	616	Tab. 672.	Command parameter table	645
Tab. 633.	ROM control - Offset: D0h	617	Tab. 673.	NVM PRINCE encryption and decryption registers	646
Tab. 634.	SRAM control - Offset: D4h	617	Tab. 674.	NPX control register - Offset: 0h	646
Tab. 635.	RAMX0 to RAM20 power modes controls - Offset: D8h	617	Tab. 675.	NPX status register - Offset: 4h	648
Tab. 636.	Bluetooth Low Energy RAM power mode controls - Offset: DCh	618	Tab. 676.	Bitmap control for memory context 0 - Offset: 10h	648
Tab. 637.	SRAM assist control 0 - Offset: E0h	619	Tab. 677.	Bitmap control for memory context 1 - Offset: 10h	649
Tab. 638.	SRAM assist control 1 - Offset: E4h	619	Tab. 678.	Bitmap control for memory context 2 - Offset: 10h	649
Tab. 639.	Design for Test - enable/disable - Offset: F8h	619	Tab. 679.	Bitmap control for memory context 3 - Offset: 10h	649
Tab. 640.	Design for Test - DCDC control - Offset: FCh	620	Tab. 680.	True random number generator registers	650
Tab. 641.	Design for test - capacitor test - Offset: 100h	621	Tab. 681.	Miscellaneous control register - Offset: 0h	652
			Tab. 682.	Statistical check miscellaneous register - Offset: 4h	654
			Tab. 683.	Poker range register - Offset: 8h	655
			Tab. 684.	Poker maximum limit register - Offset: Ch	655

Tab. 685. Poker square calculation result register - Offset: Ch 656	Tab. 718. Mask register - Offset: A8h 673
Tab. 686. Seed control register - Offset: 10h 656	Tab. 719. Interrupt status register - Offset: ACh 673
Tab. 687. Sparse bit limit register - Offset: 14h 657	Tab. 720. Version ID register (MS) - Offset: F0h 674
Tab. 688. Total samples register - Offset: 14h 657	Tab. 721. Version ID register (LS) - Offset: F4h 674
Tab. 689. Frequency count minimum limit register - Offset: 18h 658	Tab. 722. Physical unclonable function control registers 675
Tab. 690. Frequency count register - Offset: 1Ch 658	Tab. 723. PUF control register - Offset: 0h 676
Tab. 691. Frequency count maximum limit register - Offset: 1Ch 659	Tab. 724. PUF key index register - Offset: 4h 676
Tab. 692. Statistical check monobit count register - Offset: 20h 659	Tab. 725. PUF key size register - Offset: 8h 676
Tab. 693. Statistical check monobit limit register - Offset: 20h 660	Tab. 726. PUF status register - Offset: 20h 676
Tab. 694. Statistical check run length 1 count register - Offset: 24h 660	Tab. 727. PUF allow register - Offset: 28h 677
Tab. 695. Statistical check run length 1 limit register - Offset: 24h 661	Tab. 728. PUF key input register - Offset: 40h 677
Tab. 696. Statistical check run length 2 count register - Offset: 28h 661	Tab. 729. PUF code input register - Offset: 44h 677
Tab. 697. Statistical check run length 2 limit register - Offset: 28h 662	Tab. 730. PUF code output register - Offset: 48h 677
Tab. 698. Statistical check run length 3 count register - Offset: 2Ch 662	Tab. 731. PUF key output index register - Offset: 60h 677
Tab. 699. Statistical check run length 3 limit register - Offset: 2Ch 663	Tab. 732. PUF key output register - Offset: 64h 678
Tab. 700. Statistical check run length 4 count register - Offset: 30h 664	Tab. 733. PUF interface status and clear register - Offset: DCh 678
Tab. 701. Statistical check run length 4 limit register - Offset: 30h 664	Tab. 734. PUF interrupt enable - Offset: 100h 678
Tab. 702. Statistical check run length 5 count register - Offset: 34h 665	Tab. 735. PUF interrupt status - Offset: 104h 679
Tab. 703. Statistical check run length 5 limit register - Offset: 34h 665	Tab. 736. PUF config register for block bits - Offset: 10Ch 679
Tab. 704. Statistical check run length 6+ count register - Offset: 38h 666	Tab. 737. Key lock - Offset: 200h 680
Tab. 705. Statistical check run length 6+ limit register - Offset: 38h 666	Tab. 738. Key enable - Offset: 204h 681
Tab. 706. Status register - Offset: 3Ch 667	Tab. 739. Reinitialize keys shift registers counters - Offset: 208h 681
Tab. 707. Entropy read register - Offset: 40h + (a × 4h) 668	Tab. 740. Block index - Offset: 20Ch 682
Tab. 708. Statistical check poker count 1 and 0 register - Offset: 80h 669	Tab. 741. Block index duplicate - Offset: 210h 682
Tab. 709. Statistical check poker count 3 and 2 register - Offset: 84h 669	Tab. 742. Offset 683
Tab. 710. Statistical check poker count 5 and 4 register - Offset: 88h 669	Tab. 743. Key mask 683
Tab. 711. Statistical check poker count 7 and 6 register - Offset: 8Ch 670	Tab. 744. Block index status - Offset: 254h 684
Tab. 712. Statistical check poker count 9 and 8 register - Offset: 90h 670	Tab. 745. Key shift status - Offset: 258h 686
Tab. 713. Statistical check poker count B and A register - Offset: 94h 670	Tab. 746. PUF SRAM control registers 687
Tab. 714. Statistical check poker count D and C register - Offset: 98h 671	Tab. 747. Configuration register - Offset: 300h 687
Tab. 715. Statistical check poker count F and E register - Offset: 9Ch 671	Tab. 748. Status register - Offset: 304h 688
Tab. 716. Security configuration register - Offset: A0h 671	Tab. 749. SRAM configuration register - Offset: 308h 688
Tab. 717. Interrupt control register - Offset: A4h 672	Tab. 750. Interrupt enable clear register - Offset: 3D8h 688
	Tab. 751. Interrupt enable set register - Offset: 3DCh 689
	Tab. 752. Interrupt status register - Offset: 3E0h 689
	Tab. 753. Interrupt enable register - Offset: 3E4h 690
	Tab. 754. Interrupt status clear register - Offset: 3E8h 690
	Tab. 755. Interrupt status set - Offset: 3ECh 690
	Tab. 756. Module identification register - Offset: 3FCh 691
	Tab. 757. Programmable logic unit control registers 692
	Tab. 758. LUTn input y MUX - Offset: 0h + (n × 20h) + (y × 4h) 693
	Tab. 759. Truth table contents for LUTn - Offset: 800h + (n × 20h) 694
	Tab. 760. Current state of the 8 designated PLU outputs - Offset: 900h 694
	Tab. 761. Wake-up interrupt control for PLU - Offset: 904h 695
	Tab. 762. Selection of source to be connected to PLU output n - Offset: 0h + (n × 4h) 696
	Tab. 763. ROM control and patch control registers 698
	Tab. 764. ROMCP data registers - Offset: F0h + (n × -4h) 698
	Tab. 765. ROMCP control register - Offset: F4h 699

Tab. 766.	ROMCP enable register high - Offset: F8h	699	Tab. 805.	SCT output counter direction control register - Offset: 54h	726
Tab. 767.	ROMCP enable register low - Offset: FCh	700	Tab. 806.	SCT conflict resolution register - Offset: 58h	727
Tab. 768.	ROMCP address registers - Offset: 100h + (a × 4h)	700	Tab. 807.	SCT DMA request 0 register - Offset: 5Ch	730
Tab. 769.	ROMCP status registers - Offset: 208h	701	Tab. 808.	SCT DMA request 1 register - Offset: 60h	730
Tab. 770.	AHB peripherals memory map	703	Tab. 809.	SCT event interrupt enable register - Offset: F0h	731
Tab. 771.	DMA engine registers	704	Tab. 810.	SCT event flag register- Offset: F4h	731
Tab. 772.	DMA control - Offset: 0h	705	Tab. 811.	SCT conflict interrupt enable register - Offset: F8h	731
Tab. 773.	Interrupt status - Offset: 4h	705	Tab. 812.	SCT conflict flag register - Offset: FCh	731
Tab. 774.	SRAM address of the channel configuration table - Offset: 8h	706	Tab. 813.	SCT capture register of capture channel n - Offset: 100h + (n × 4h)	732
Tab. 775.	Channel enable read and set for all DMA channels - Offset: 20h	706	Tab. 814.	SCT match value register of match channel n - Offset: 100h + (n × 4h)	732
Tab. 776.	Channel enable clear for all DMA channels - Offset: 28h	706	Tab. 815.	SCT capture control register n - Offset: 200h + (n × 4h)	732
Tab. 777.	Channel active status for all DMA channels - Offset: 30h	706	Tab. 816.	SCT match reload value register n - Offset: 200h + (n × 4h)	733
Tab. 778.	Channel busy status for all DMA channels - Offset: 38h	707	Tab. 817.	SCT event state register n - Offset: 300h + (n × 4h)	733
Tab. 779.	Error interrupt status for all DMA channels - Offset: 40h	707	Tab. 818.	SCT event control register n - Offset: 304h + (n × 4h)	733
Tab. 780.	Interrupt enable read and set for all DMA channels - Offset: 48h	707	Tab. 819.	SCT output n set register - Offset: 500h + (n × 4h)	735
Tab. 781.	Interrupt enable clear for all DMA channels - Offset: 50h	707	Tab. 820.	SCT output n clear register - Offset: 504h + (n × 4h)	735
Tab. 782.	Interrupt A status for all DMA channels - Offset: 58h	708	Tab. 821.	Flexcomm serial communication registers	736
Tab. 783.	Interrupt B status for all DMA channels - Offset: 60h	708	Tab. 822.	Peripheral select and flexcomm ID register - Offset: FF8h	736
Tab. 784.	Set ValidPending control bits for all DMA channels - Offset: 68h	708	Tab. 823.	Peripheral identification register - Offset: FFCh	737
Tab. 785.	Set trigger control bits for all DMA channels - Offset: 70h	708	Tab. 824.	Flexcomm serial communication registers	738
Tab. 786.	Channel abort control for all DMA channels - Offset: 78h	709	Tab. 825.	USART configuration register - Offset: 0h	739
Tab. 787.	Register reset values	709	Tab. 826.	USART control register - Offset: 4h	743
Tab. 788.	Configuration register for DMA channel n - Offset: 400h + (n × 10h)	709	Tab. 827.	USART status register - Offset: 8h	744
Tab. 789.	Register reset values	712	Tab. 828.	Interrupt enable read and set register for USART (not FIFO) status - Offset: Ch	746
Tab. 790.	Control and status register for DMA channel n - Offset: 404h + (n × 10h)	713	Tab. 829.	Interrupt enable clear register - Offset: 10h ...	746
Tab. 791.	Register reset values	713	Tab. 830.	Baud rate generator register - Offset: 20h	747
Tab. 792.	Transfer configuration register for DMA channel n - Offset: 408h + (n × 10h)	713	Tab. 831.	Interrupt status register - Offset: 24h	747
Tab. 793.	State configurable timer/pulse width modulator registers	717	Tab. 832.	Oversample selection register for asynchronous communication - Offset: 28h ..	748
Tab. 794.	SCT configuration register - Offset: 0h	718	Tab. 833.	Address register for automatic address matching - Offset: 2Ch	748
Tab. 795.	SCT control register - Offset: 4h	720	Tab. 834.	FIFO configuration and enable register - Offset: E00h	749
Tab. 796.	SCT limit event select register - Offset: 8h	722	Tab. 835.	FIFO status register - Offset: E04h	750
Tab. 797.	SCT halt event select register - Offset: Ch	722	Tab. 836.	FIFO trigger settings for interrupt and DMA request - Offset: E08h	751
Tab. 798.	SCT stop event select register - Offset: 10h ..	722	Tab. 837.	FIFO interrupt enable set (enable) and read register - Offset: E10h	753
Tab. 799.	SCT start event select register - Offset: 14h ..	723	Tab. 838.	FIFO interrupt enable clear (disable) and read register - Offset: E14h	753
Tab. 800.	SCT counter register - Offset: 40h	723	Tab. 839.	FIFO interrupt status register - Offset: E18h ..	754
Tab. 801.	SCT state register - Offset: 44h	723	Tab. 840.	FIFO write data - Offset: E20h	754
Tab. 802.	SCT input register - Offset: 48h	723	Tab. 841.	FIFO read data - Offset: E30h	754
Tab. 803.	SCT match/capture mode register - Offset: 4Ch	725			
Tab. 804.	SCT output register - Offset: 50h	726			

Tab. 842.	FIFO data read with no FIFO pop - Offset: E40h	755	Tab. 882.	Word pin registers for all port GPIO pins - Offset: 1000h + (n × 4h)	791
Tab. 843.	FIFO size register - Offset: E48h	755	Tab. 883.	Direction registers for all port GPIO pins - Offset: 2000h	791
Tab. 844.	Serial peripheral interface registers	756	Tab. 884.	Mask register for all port GPIO pins - Offset: 2080h	792
Tab. 845.	SPI configuration register - Offset: 400h	757	Tab. 885.	Port pin register for all port GPIO pins - Offset: 2100h	792
Tab. 846.	SPI delay register - Offset: 404h	758	Tab. 886.	Masked port register for all port GPIO pins - Offset: 2180h	792
Tab. 847.	SPI status - Offset: 408h	759	Tab. 887.	Write: Set register for port; Read: output bits for port - Offset: 2200h	793
Tab. 848.	SPI interrupt enable read and set - Offset: 40Ch	760	Tab. 888.	Clear port for all port GPIO pins - Offset: 2280h	793
Tab. 849.	SPI interrupt enable clear - Offset: 410h	760	Tab. 889.	Toggle port for all port GPIO pins - Offset: 2300h	793
Tab. 850.	SPI clock divider - Offset: 424h	761	Tab. 890.	Set pin direction bits for port - Offset: 2380h	793
Tab. 851.	SPI interrupt status - Offset: 428h	761	Tab. 891.	Clear pin direction bits for port - Offset: 2400h	794
Tab. 852.	FIFO configuration and enable register - Offset: E00h	762	Tab. 892.	Toggle pin direction bits for port - Offset: 2480h	794
Tab. 853.	FIFO status register - Offset: E04h	763	Tab. 893.	CRC engine registers	795
Tab. 854.	FIFO trigger settings for interrupt and DMA request - Offset: E08h	764	Tab. 894.	CRC mode register - Offset: 0h	795
Tab. 855.	FIFO interrupt enable set (enable) and read register - Offset: E10h	766	Tab. 895.	CRC seed register - Offset: 4h	796
Tab. 856.	FIFO interrupt enable clear (disable) and read register - Offset: E14h	766	Tab. 896.	CRC checksum register - Offset: 8h	796
Tab. 857.	FIFO interrupt status register - Offset: E18h	767	Tab. 897.	CRC data register - Offset: 8h	796
Tab. 858.	FIFO write data - Offset: E20h	767	Tab. 898.	MCU debugger mailbox engine registers	797
Tab. 859.	FIFO read data - Offset: E30h	767	Tab. 899.	Debug mailbox control and status register - Offset: 0h	797
Tab. 860.	FIFO data read with no FIFO pop - Offset: E40h	768	Tab. 900.	Debug mailbox request register - Offset: 4h	798
Tab. 861.	FIFO size register - Offset: E48h	768	Tab. 901.	Debug mailbox return value register - Offset: 8h	798
Tab. 862.	I2C registers	769	Tab. 902.	Identification register - Offset: FCh	798
Tab. 863.	Configuration for shared functions - Offset: 800h	770	Tab. 903.	SPI flash interface registers	799
Tab. 864.	Status register for master, slave, and monitor functions - Offset: 804h	771	Tab. 904.	SPIFI control register - Offset: 0h	800
Tab. 865.	Interrupt enable set and read register - Offset: 808h	777	Tab. 905.	SPIFI command register - Offset: 4h	802
Tab. 866.	Interrupt enable clear register - Offset: 80Ch	779	Tab. 906.	SPIFI address register - Offset: 8h	803
Tab. 867.	Timeout value register - Offset: 810h	780	Tab. 907.	SPIFI intermediate data register - Offset: Ch	803
Tab. 868.	Clock predivider for the entire I2C interface - Offset: 814h	781	Tab. 908.	SPIFI limit register - Offset: 10h	803
Tab. 869.	Interrupt status register for master, slave, and monitor functions - Offset: 818h	781	Tab. 909.	SPIFI data register - Offset: 14h	803
Tab. 870.	Master control register - Offset: 820h	782	Tab. 910.	SPIFI memory command register - Offset: 18h	804
Tab. 871.	Master timing configuration - Offset: 824h	783	Tab. 911.	SPIFI status register - Offset: 1Ch	805
Tab. 872.	Combined master receiver and transmitter data register - Offset: 828h	785	Tab. 912.	Code watchdog registers	806
Tab. 873.	Slave control register - Offset: 840h	785	Tab. 913.	Code watchdog control - Offset: 0h	807
Tab. 874.	Combined slave receiver and transmitter data register - Offset: 844h	786	Tab. 914.	Instruction timer reload - Offset: 4h	808
Tab. 875.	Slave address register - Offset: 848h	787	Tab. 915.	Instruction timer - Offset: 8h	808
Tab. 876.	Offset	788	Tab. 916.	Secure counter - Offset: Ch	809
Tab. 877.	Slave address register	788	Tab. 917.	Status register 1 - Offset: 10h	809
Tab. 878.	Slave qualification for address 0 - Offset: 858h	788	Tab. 918.	Status register 2 - Offset: 14h	809
Tab. 879.	Monitor receiver data register - Offset: 880h	789	Tab. 919.	Hardware flags - Offset: 18h	810
Tab. 880.	General purpose I/O registers	790	Tab. 920.	Persistent data storage - Offset: 1Ch	811
Tab. 881.	Byte pin registers for all port GPIO pins - Offset: 0h + (n × 1h)	791	Tab. 921.	Write address for issuing the START command - Offset: 20h	811
			Tab. 922.	Write address for issuing the STOP command - Offset: 24h	811

Tab. 923. Write address for issuing the RESTART command - Offset: 28h	811	Tab. 968. Result register 3 - Offset: 3Ch	830
Tab. 924. Write address for issuing the ADD command - Offset: 2Ch	811	Tab. 969. Optional mask register - Offset: 60h	830
Tab. 925. Write address for issuing the ADD1 command - Offset: 30h	812	Tab. 970. Optional remask register - Offset: 64h	831
Tab. 926. Write address for issuing the ADD16 command - Offset: 34h	812	Tab. 971. Security lock register - Offset: 80h	831
Tab. 927. Write address for issuing the ADD256 command - Offset: 38h	812	Tab. 972. AHB secure controller registers	832
Tab. 928. Write address for issuing the SUB command - Offset: 3Ch	812	Tab. 973. Security access rules for flash and ROM slaves - Offset: 0h	834
Tab. 929. Write address for issuing the SUB1 command - Offset: 40h	812	Tab. 974. Offset	835
Tab. 930. Write address for issuing the SUB16 command - Offset: 44h	813	Tab. 975. Security access rules for FLASH sector 0 to sector 16	835
Tab. 931. Write address for issuing the SUB256 command - Offset: 48h	813	Tab. 976. Offset	838
Tab. 932. Write address for issuing the ASSERT16 command - Offset: 4Ch	813	Tab. 977. Security access rules for ROM sector 0 to sector 31	839
Tab. 933. HASHAES peripheral registers	814	Tab. 978. Security access rules for RAMX slave - Offset: 30h	840
Tab. 934. Hash control register (CTRL) - Offset: 0h	815	Tab. 979. Security access rules for RAMX memory - Offset: 40h	841
Tab. 935. Status of hash peripheral - Offset: 4h	816	Tab. 980. Security access rules for RAM0 slave - Offset: 50h	842
Tab. 936. Interrupt enable set register - Offset: 8h	818	Tab. 981. Security access rules for RAM0 memory - Offset: 60h	842
Tab. 937. Interrupt enable clear register - Offset: Ch	818	Tab. 982. Security access rules for RAM1 slave - Offset: 70h	843
Tab. 938. Memory access control - Offset: 10h	819	Tab. 983. Security access rules for RAM0 memory - Offset: 60h	844
Tab. 939. Start address for memory access - Offset: 14h	819	Tab. 984. Security access rules for RAM2 slave - Offset: 90h	846
Tab. 940. 16 words input data - Offset: 20h	819	Tab. 985. Offset	846
Tab. 941. Offset	820	Tab. 986. Security access rules for RAM0 memory	847
Tab. 942. Aliases for input data	820	Tab. 987. Security access rules for SPIFI slave - Offset: B0h	848
Tab. 943. Result digest - Offset: 40h + (a × 4h)	820	Tab. 988. Offset	849
Tab. 944. Result digest - Offset: 60h + (a × 4h)	820	Tab. 989. Security access rules for SPIFI memory	850
Tab. 945. Crypto configuration - Offset: 80h	821	Tab. 990. Security access rules for both APB bridge slaves - Offset: D0h	852
Tab. 946. Hardware configuration - Offset: 84h	822	Tab. 991. Security access rules for APB bridge 0 peripherals - Offset: E0h	853
Tab. 947. Register lock - Offset: 8Ch	823	Tab. 992. Security access rules for APB bridge 0 peripherals - Offset: E4h	855
Tab. 948. Offset	823	Tab. 993. Security access rules for APB bridge 0 peripherals - Offset: E8h	856
Tab. 949. ICB mask	823	Tab. 994. Security access rules for APB Bridge 0 peripherals - Offset: ECh	856
Tab. 950. SHA digest to reload - Offset: A0h + (a × 4h)	823	Tab. 995. Security access rules for APB bridge 1 peripherals - Offset: F0h	857
Tab. 951. Pseudo random number generator seed value - Offset: D0h	824	Tab. 996. Security access rules for APB bridge 1 peripherals - Offset: F4h	857
Tab. 952. Pseudo random number - Offset: D8h	824	Tab. 997. Security access rules for APB Bridge 1 peripherals - Offset: F8h	858
Tab. 953. Crypto-coprocessor registers	825	Tab. 998. Security access rules for APB bridge 1 peripherals - Offset: FCh	859
Tab. 954. Offsets of AB and CD in the RAM - Offset: 0h	826	Tab. 999. Security access rules for AHB peripherals on port 7 - slave 0 - Offset: 100h	861
Tab. 955. Control 1 register - Offset: 4h	826	Tab. 1000. Security access rules for AHB peripherals on port 7 - slave 1 - Offset: 104h	862
Tab. 956. Loader register - Offset: 8h	827	Tab. 1001. Security access rules for AHB peripherals on port 8 - slave 0 - Offset: 110h	863
Tab. 957. Status register - Offset: Ch	828	Tab. 1002. Security access rules for AHB peripherals on port 8 - slave 1 - Offset: 114h	863
Tab. 958. Interrupt enable set register - Offset: 10h	828		
Tab. 959. Interrupt enable clear register - Offset: 14h	828		
Tab. 960. Interrupt status bits - Offset: 18h	829		
Tab. 961. A register - Offset: 20h	829		
Tab. 962. B register - Offset: 24h	829		
Tab. 963. C register - Offset: 28h	829		
Tab. 964. D register - Offset: 28h	829		
Tab. 965. Result register 0 - Offset: 30h	830		
Tab. 966. Result register 1 - Offset: 34h	830		
Tab. 967. Result register 2 - Offset: 38h	830		

Tab. 1003. Security access rules for AHB peripherals on port 9 - slave 0 - Offset: 120h	864	Tab. 1022. Interrupt set pending register - Offset: 100h + (a × 4h)	888
Tab. 1004. Security access rules for AHB peripherals on port 9 - slave 1 - Offset: 124h	865	Tab. 1023. Interrupt clear pending register - Offset: 180h + (a × 4h)	888
Tab. 1005. Security access rules for AHB_SEC_CTRL memory - Offset: 130h	865	Tab. 1024. Interrupt active bit register - Offset: 200h + (a × 4h)	889
Tab. 1006. Security access rules for AHB peripherals on port 10 - slave 0 - Offset: 140h	867	Tab. 1025. Interrupt target non-secure register - Offset: 280h + (a × 4h)	889
Tab. 1007. Most recent security violation address for AHB layer n - Offset: E00h + (a × 4h)	867	Tab. 1026. Interrupt target non-secure register - Offset: 300h + (a × 4h)	889
Tab. 1008. Most recent security violation miscellaneous information for AHB layer n - Offset: E80h + (a × 4h)	868	Tab. 1027. Software trigger interrupt register - Offset: E00h	890
Tab. 1009. Most recent security violation miscellaneous information for AHB layer n - Offset: E80h + (a × 4h)	868	Tab. 1028. System control block registers	890
Tab. 1010. Security violation address/information registers valid flags - Offset: F00h	870	Tab. 1029. Application interrupt and reset control register - Offset: Ch	891
Tab. 1011. Secure GPIO mask for port 0 pins - Offset: F80h	871	Tab. 1030. SCR control of features of entry to and exit from low-power state - Offset: 10h	892
Tab. 1012. Security general-purpose register access control - Offset: FBCh	874	Tab. 1031. System handler control and state register - Offset: 24h	893
Tab. 1013. Master secure level register - Offset: FD0h	874	Tab. 1032. Non-secure access control register - Offset: 8Ch	896
Tab. 1014. Master secure level anti-pole register - Offset: FD4h	877	Tab. 1033. Security attribution unit registers	898
Tab. 1015. Miscellaneous control signals for in Cortex M33 (CPU0) - Offset: FECh	878	Tab. 1034. Security attribution unit control register - Offset: D0h	899
Tab. 1016. Secure control duplicate register - Offset: FF8h	881	Tab. 1035. Security attribution unit type register - Offset: D4h	899
Tab. 1017. Secure control register - Offset: FFCh	882	Tab. 1036. Security attribution unit region number register - Offset: D8h	900
Tab. 1018. Arm NVIC and TrustZone	886	Tab. 1037. Security attribution unit region base address register - Offset: DCh	900
Tab. 1019. Nested vector interrupt control registers	886	Tab. 1038. Security attribution unit region limit address register - Offset: E0h	901
Tab. 1020. Interrupt set enable register - Offset: 0h + (a × 4h)	887	Tab. 1039. Secure fault status register - Offset: E4h	901
Tab. 1021. Interrupt clear enable register - Offset: 80h + (a × 4h)	887	Tab. 1040. Secure fault address register - Offset: E8h	903
		Tab. 1041. Abbreviations	905
		Tab. 1042. Revision history	907

Figures

Fig. 1. MCX W23 block diagram	6	Fig. 18. Command with no data phase	79
Fig. 2. MCX W23 Clock generation (Part 1 of 2)	21	Fig. 19. Packet flow command with incoming data phase	80
Fig. 3. MCX W23 Clock generation (Part 2 of 2)	22	Fig. 20. Command with outgoing data phase	81
Fig. 4. Simplified block diagram of the flash accelerator	24	Fig. 21. Ping packet protocol sequence	85
Fig. 5. Frequency measure block diagram	26	Fig. 22. Protocol sequence for GetProperty command	91
Fig. 6. Block diagram	28	Fig. 23. Protocol sequence for SetProperty Command	93
Fig. 7. MCX W23 boot flowchart	34	Fig. 24. Protocol sequence for FlashEraseAll command	94
Fig. 8. Secure boot ROM flowchart	42	Fig. 25. Protocol sequence for FlashEraseRegion command	96
Fig. 9. Protected Flash Region	43	Fig. 26. Command sequence for ReadMemory	97
Fig. 10. RoTKH generation process	48	Fig. 27. Protocol sequence for WriteMemory command	99
Fig. 11. Key store area in PFR	51	Fig. 28. Protocol sequence for FillMemory command	101
Fig. 12. Structure of unsigned CRC images	54	Fig. 29. Protocol sequence for reset command	103
Fig. 13. Structure of signed images	55		
Fig. 14. Structure of certificate block	57		
Fig. 15. Location of TrustZone configuration data in the image file	65		
Fig. 16. Signed image preparation	68		
Fig. 17. Reserved RAM region for the boot ROM	78		

Fig. 30.	Protocol sequence for ConfigureMemory command	104	Fig. 76.	Micro-tick timer block diagram	241
Fig. 31.	Host reads an ACK from target via UART	117	Fig. 77.	MRT block diagram	243
Fig. 32.	Host reads a ping response from target via UART	117	Fig. 78.	RTC clocking and block diagram	246
Fig. 33.	Host reads a command response from target via UART	118	Fig. 79.	System tick timer block diagram	249
Fig. 34.	Host reads ping response from target via I2C	119	Fig. 80.	WWDT clocking	252
Fig. 35.	Host reads ACK packet from target via I2C	119	Fig. 81.	Windowed watchdog timer block diagram	254
Fig. 36.	Host reads response from target via I2C	120	Fig. 82.	Early watchdog feed with windowed mode enabled	255
Fig. 37.	Physical interface for SPI ISP	121	Fig. 83.	Correct watchdog feed with windowed mode enabled	256
Fig. 38.	Host reads ping packet from target via SPI	121	Fig. 84.	Watchdog warning interrupt	256
Fig. 39.	Host reads ACK from target via SPI	122	Fig. 85.	OS event timer block diagram	262
Fig. 40.	Host reads response from target via SPI	122	Fig. 86.	Flexcomm interface block diagram	264
Fig. 41.	ROM API structure	125	Fig. 87.	I2C block diagram	273
Fig. 42.	Frequency measure block diagram	146	Fig. 88.	Select clock for FCLK	280
Fig. 43.	Standard GPIO pin configuration	182	Fig. 89.	USART block diagram	284
Fig. 44.	Combo I2C/GPIO pin configuration	182	Fig. 90.	Hardware flow control using RTS and CTS	286
Fig. 45.	Open drain mode	184	Fig. 91.	SPI block diagram	293
Fig. 46.	Generic input multiplexing	191	Fig. 92.	Basic SPI operating mode	294
Fig. 47.	SCT0 input multiplexing	191	Fig. 93.	Pre_delay and Post_delay	295
Fig. 48.	Pin interrupt multiplexing	192	Fig. 94.	Frame_delay	296
Fig. 49.	Pin interrupt secure multiplexing	192	Fig. 95.	Transfer_delay	297
Fig. 50.	DMA trigger input multiplexing	193	Fig. 96.	Examples of data stalls	300
Fig. 51.	Pin interrupts	195	Fig. 97.	Opcode-only commands	303
Fig. 52.	Pattern match engine connections	197	Fig. 98.	Read commands	304
Fig. 53.	Pattern match bit slice with detect logic	198	Fig. 99.	PLU block diagram	309
Fig. 54.	Pattern match engine examples: sticky edge detect	200	Fig. 100.	PLU register/multiplexing detail	310
Fig. 55.	Pattern match engine examples: Windowed non-sticky edge detect evaluates as true	201	Fig. 101.	CRC block diagram	313
Fig. 56.	Pattern match engine examples: Windowed non-sticky edge detect evaluates as false	201	Fig. 102.	Arm TrustZone system security abstract view	316
Fig. 57.	Secure pin interrupt connections	203	Fig. 103.	Cortex-M4 (or M3) vs Cortex-M33	317
Fig. 58.	Pattern match engine connections	204	Fig. 104.	Secure state and Nonsecure state view for TrustZone for ARMv8-M	318
Fig. 59.	Secure pattern match bit slice with detect logic	205	Fig. 105.	Connection of DAU to CPU0	319
Fig. 60.	DMA block diagram	211	Fig. 106.	IDAU implementation	320
Fig. 61.	Interleaved transfer in a single buffer	217	Fig. 107.	Security attribute definition as a combination of SAU and IDAU	321
Fig. 62.	SCTimer/PWM clocking	220	Fig. 108.	Program memory space aliasing and security attribution example	322
Fig. 63.	SCTimer/PWM connections	220	Fig. 109.	Test target instruction usage	323
Fig. 64.	SCTimer/PWM block diagram	224	Fig. 110.	Security tiers as defined using HPRIV and HNONSEC sideband signals	323
Fig. 65.	SCTimer/PWM counter and select logic	225	Fig. 111.	System view with secure AHB bus	324
Fig. 66.	Match logic	225	Fig. 112.	Security system	327
Fig. 67.	Capture logic	226	Fig. 113.	Key storage	328
Fig. 68.	Event selection	226	Fig. 114.	Possible flows, states, and actions	333
Fig. 69.	Output slice i	227	Fig. 115.	KC header format	335
Fig. 70.	SCT interrupt generation	228	Fig. 116.	Key byte order on the Key interface for 128-bit key	336
Fig. 71.	SCT configuration example	232	Fig. 117.	2x512-bit buffers are used for AES	343
Fig. 72.	32-bit counter/timer block diagram	237	Fig. 118.	SHA input data block	345
Fig. 73.	A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled	239	Fig. 119.	Conceptual diagram of flash crypto regions	347
Fig. 74.	A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled	239	Fig. 120.	CASPER block diagram	349
Fig. 75.	Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.	240	Fig. 121.	CASPER memory map	350
			Fig. 122.	CASPER example	351
			Fig. 123.	Internal debug system elements and connections	355

Fig. 124. Debug authentication protocol usage example	366	Fig. 128. Debug authentication response (DAR) fields	372
Fig. 125. Debug authentication flow	367	Fig. 129. TX power steps in TXM1 mode	389
Fig. 126. Debug Credential certificate fields	368	Fig. 130. TX power steps in TXM2 mode	389
Fig. 127. Debug authentication Challenge (DAC) fields	371		

Contents

1	Introduction	2	6.3.2	ISP boot mode	35
1.1	Features	2	6.3.3	SPI flash recovery	35
1.2	Block diagram	6	6.4	PFR region definitions	36
1.3	Architectural overview	7	6.5	MbedTLS ROM library	36
1.4	Arm Cortex-M33 with TrustZone	7	7	Secure boot ROM	40
1.5	Arm Cortex-M33-integrated floating point unit (FPU)	7	7.1	Function description	40
1.6	On-chip static RAM	7	7.1.1	Secure boot	40
1.7	Ordering information	8	7.1.2	Secure firmware update	40
1.7.1	Ordering options	8	7.1.3	Extending the chain of trust	40
2	Memory map	10	7.1.4	Miscellaneous functions	41
2.1	AHB multilayer matrix	10	7.1.5	Boot flow diagram	41
2.2	Memory protection unit (MPU)	10	7.2	Data structures	42
2.3	TrustZone and system mapping on this device	10	7.2.1	Overview	42
2.4	Memory map overview	11	7.2.2	Secure boot related configuration fields in PFR	44
2.5	APB peripherals	12	7.2.2.1	Customer manufacturing/factory programmable area (CMPA) page	44
2.6	AHB peripherals	13	7.2.2.2	Customer field programmable area (CFPA) page	49
2.7	SRAM	14	7.2.3	Key storage in the protected flash region	50
3	Nested vectored interrupt controller (NVIC)	15	7.3	Keys	51
3.1	Features	15	7.3.1	PUF key code format	52
3.2	General description	15	7.3.2	Key descriptions	53
3.2.1	Interrupt sources	15	7.3.3	Plain image structure	54
4	System Control (SYSCON)	19	7.3.4	Signed image structure	54
4.1	Features	19	7.3.5	Certificate block	56
4.2	Basic configuration	19	7.3.5.1	Certificate block header	57
4.2.1	Configure the main clock and system clock	19	7.3.5.2	Certificate table	58
4.2.2	Measure the frequency of a clock signal	19	7.3.6	ROM firmware update using SB file	59
4.3	Pin description	20	7.3.6.1	Header	59
4.4	Clock generation	20	7.3.6.2	MAC of the section MAC table	59
4.5	Functional description	23	7.3.6.3	Key blob	59
4.5.1	Reset	23	7.3.6.4	Sections	59
4.5.2	Clock	23	7.3.6.5	Usage of firmware update	61
4.5.3	Brownout detection	23	7.4	ROM TrustZone support	61
4.5.4	Start-up behavior	24	7.4.1	TrustZone image type	61
4.5.5	Flash accelerator functional description	24	7.4.1.1	TrustZone disabled image	61
4.5.5.1	Flash memory bank	25	7.4.1.2	TrustZone enabled image	62
4.5.5.2	Flash programming constraints	25	7.4.2	TrustZone preset data	62
4.5.6	Frequency measure function	25	7.4.2.1	TrustZone preset data structure	63
5	Flash	27	7.4.2.2	TrustZone image type restriction control during boot process	66
5.1	General description	27	7.4.3	Boot ROM API and TrustZone	66
5.2	Features	27	7.4.3.1	TrustZone disabled images	66
5.3	Block diagram	27	7.4.3.2	TrustZone enabled images	66
5.4	Usage	29	7.5	Secure boot usage	67
5.4.1	Initialization	29	7.5.1	Keys and certificates	67
5.4.2	Erasing	29	7.5.1.1	CFPA/CMPA page preparation	67
5.4.3	Programming	29	7.5.1.2	Signed image preparation	68
5.4.4	Reading	30	7.5.1.3	Loading signed image	68
5.4.5	Bit errors	30	7.5.2	Internal flash encryption using PRINCE engine	68
5.4.6	Caching	30	7.5.2.1	PRINCE-related PUF key store setup	69
6	Boot ROM	31	7.5.2.2	PRINCE region configuration with blhost	69
6.1	Features	31	7.5.2.3	Upload image	70
6.2	General description	31	7.6	Secure ROM API	71
6.3	Boot modes	35			
6.3.1	Passive boot mode	35			

7.6.1	Image authentication API	71	8.6.5	UART ISP commands	118
7.6.1.1	kb_authenticate	71	8.7	I2C in-system programming	118
7.6.1.2	hashcrypt_irq_handler	72	8.7.1	Introduction	118
7.6.2	PRINCE ROM API	72	8.7.2	I2C ISP command format	120
7.6.2.1	bus_crypto_engine_set_encrypt_for_		8.7.3	I2C ISP response format	120
	address_range	73	8.7.4	I2C ISP data format	120
7.6.3	SB file processing API	74	8.7.5	I2C ISP commands	120
7.6.3.1	kb_init	74	8.8	SPI in-system programming	120
7.6.3.2	kb_deinit	75	8.8.1	Introduction	120
7.6.3.3	kb_execute	75	8.8.2	SPI ISP command format	123
8	In-system programming (ISP) and in-		8.8.3	SPI ISP response format	123
	application programming (IAP)	77	8.8.4	SPI ISP data format	123
8.1	Features	77	8.8.5	SPI ISP commands	123
8.2	General description	77	9	Flash API	124
8.2.1	Bootloader	77	9.1	Features	124
8.2.2	In-system programming (ISP) and in-		9.2	General description	124
	application programming (IAP)	77	9.2.1	ROM API structure	124
8.2.3	Memory map after any reset	78	9.2.2	Flash APIs	126
8.2.4	Interrupts and SRAM usage	78	9.2.2.1	flash_init	127
8.2.4.1	Interrupts during IAP	78	9.2.2.2	flash_erase	127
8.2.4.2	RAM usage	78	9.2.2.3	flash_program	128
8.3	In-system programming protocol	78	9.2.2.4	flash_verify_erase	128
8.3.1	Command with no data phase	79	9.2.2.5	flash_verify_program	129
8.3.2	Command with incoming data phase	79	9.2.2.6	flash_get_property	129
8.3.3	Command with outgoing data phase	80	9.2.3	PFR APIs	130
8.4	Bootloader packet types	82	9.2.3.1	ffr_init	130
8.4.1	Introduction	82	9.2.3.2	ffr_lock_all	131
8.4.2	Framing packet	82	9.2.3.3	ffr_cust_factory_page_write	131
8.4.3	CRC16 algorithm	84	9.2.3.4	ffr_get_uuid	132
8.4.4	Ping packet	84	9.2.3.5	ffr_get_customer_data	132
8.4.5	Ping response packet	85	9.2.3.6	ffr_keystore_write	132
8.4.6	Command packet	86	9.2.3.7	ffr_keystore_get_ac	133
8.4.7	Response packet	88	9.2.3.8	ffr_keystore_get_kc	133
8.4.8	Data packet	89	9.2.3.9	ffr_infield_page_write	134
8.4.9	ACK packet	89	9.2.3.10	ffr_get_customer_infield_data	134
8.5	Bootloader command set	90	9.2.4	runBootloader API	135
8.5.1	Introduction	90	9.3	Driver status codes	136
8.5.2	GetProperty command	90	10	Protected flash region	137
8.5.3	SetProperty command	92	10.1	Customer manufacturing programmable	
8.5.4	FlashEraseAll command	94		area (CMPA)	137
8.5.5	FlashEraseRegion command	95	10.2	Customer key store	138
8.5.6	ReadMemory command	97	10.3	Customer field programmable area (CFPA) ...	138
8.5.7	WriteMemory command	98	10.4	NXP programmed area	139
8.5.8	FillMemory command	101	10.5	Region SHA256 hash digest	139
8.5.9	Execute command	102	10.6	Customer development lifecycle state	140
8.5.10	Reset command	103	11	Analog control (ANACTRL)	145
8.5.11	ConfigureMemory command	104	11.1	Features	145
8.5.11.1	Supported memory IDs	105	11.2	Basic configuration	145
8.5.11.2	1-bit SPI NOR flash support	105	11.2.1	Measure the frequency of a clock signal	145
8.5.12	ReceiveSBFile command	106	11.3	Pin description	145
8.5.13	KeyProvisioning command	107	11.4	Function description	145
8.5.14	Get/SetProperty command properties	109	11.4.1	Frequency measure function	145
8.5.15	Bootloader command and response tags	114	11.4.1.1	Accuracy	147
8.5.16	Bootloader status error codes	114	12	Crystal oscillator (XTAL32M - XTAL32K)	148
8.6	UART ISP	116	12.1	Features	148
8.6.1	Introduction	116	12.2	Crystal oscillator API description	148
8.6.2	UART ISP command format	118	13	Power management	150
8.6.3	UART ISP response format	118	13.1	General description	150
8.6.4	UART ISP data format	118	13.1.1	Power supplies	150

13.1.2	Power domains	150	15.1	Features	181
13.1.3	Power modes	153	15.2	Basic configuration	181
13.1.4	Peripheral configuration in reduced power modes	154	15.3	General description	181
13.1.5	Wake-up process	155	15.3.1	Pin configuration	181
13.2	Functional description	155	15.3.2	IOCON registers	182
13.2.1	Power management	155	15.3.2.1	Pin function	183
13.2.2	Active mode	156	15.3.2.2	Pin mode	183
13.2.2.1	Power configuration in active mode	156	15.3.2.3	Hysteresis	183
13.2.3	Sleep mode	157	15.3.2.4	Invert pin	183
13.2.3.1	Power configuration in sleep mode	157	15.3.2.5	Analog/digital mode	183
13.2.3.2	Programming sleep mode	157	15.3.2.6	Input filter	183
13.2.3.3	Wake-up from sleep mode	157	15.3.2.7	Output slew rate	184
13.2.4	Deep-sleep mode	157	15.3.2.8	I2C modes	184
13.2.4.1	Power configuration in deep-sleep mode	158	15.3.2.9	Open-drain mode	184
13.2.4.2	Programming deep-sleep mode	159	16	General-purpose I/O (GPIO)	185
13.2.4.3	Wake-up from deep-sleep mode	159	16.1	Features	185
13.2.5	Power-down mode	159	16.2	Basic configuration	185
13.2.5.1	Power configuration in power-down mode	159	16.3	General description	185
13.2.5.2	Programming power-down mode	159	16.4	Functional description	185
13.2.5.3	Wake-up from power-down mode	160	16.4.1	Reading pin state	185
13.2.6	Deep power-down mode	160	16.4.2	GPIO output	186
13.2.6.1	Power configuration in deep power-down mode	160	16.4.3	Masked I/O	186
13.2.6.2	Programming deep-power down mode using the RTC for wake-up	160	16.4.4	GPIO direction	186
13.2.6.3	Programming deep-power down mode using the OS event timer for wake-up	160	16.4.5	Recommended practices	187
13.2.6.4	Programming deep-power down mode using the wake-up pin for wake-up	161	17	Secure general-purpose I/O (SECGPIO)	188
13.2.6.5	Wake-up from deep power-down mode	161	17.1	Features	188
13.2.7	Power-off mode	161	17.2	Basic configuration	188
14	Power profiles/power control API	162	17.3	General description	188
14.1	Features	162	17.4	Functional description	188
14.2	General description	162	17.4.1	Reading pin state	188
14.3	Power-related API descriptions	162	17.4.2	Secure GPIO output	188
14.3.1	POWER_PeripheralPowerOff/On	169	17.4.3	Masked I/O	189
14.3.2	POWER_Dis/EnableSRAM	169	17.4.4	Secure GPIO direction	189
14.3.3	POWER_Bod	170	17.4.5	Recommended practices	189
14.3.4	POWER_DcDc	171	18	Input multiplexing (INPUTMUX)	190
14.3.5	POWER_XTAL32K	173	18.1	Features	190
14.3.6	POWER_XTAL32M	174	18.2	Basic configuration	190
14.3.7	POWER_EnterSleep	175	18.3	Pin description	190
14.3.8	POWER_EnterDeepSleep	175	18.4	General description	190
14.3.8.1	Param0: exclude_from_pd	176	18.4.1	SCT0 input multiplexing	191
14.3.8.2	Param1: wakeup_interrupts	176	18.4.2	Pin interrupt input multiplexing	191
14.3.9	POWER_EnterPowerDown	176	18.4.3	Pin interrupt secure input multiplexing	192
14.3.9.1	Param0: exclude_from_pd	177	18.4.4	DMA trigger input multiplexing	192
14.3.9.2	Param1: wakeup_interrupts	177	19	Pin interrupt and pattern match (PINT)	194
14.3.9.3	Param2: cpu_retention_ctrl	178	19.1	Features	194
14.3.10	POWER_EnterDeepPowerDown	178	19.2	Basic configuration	194
14.3.10.1	Param0: exclude_from_pd	179	19.2.1	Configure pins as pin interrupts or as inputs to the pattern match engine	194
14.3.10.2	Param1: wakeup_interrupts	179	19.3	Pin description	195
14.3.10.3	Param2: wakeup_io_ctrl	179	19.4	General description	195
14.3.11	POWER_EnterPowerOff	179	19.4.1	Pin interrupts	195
14.3.11.1	Param0: exclude_from_pd	180	19.4.2	Pattern match engine	195
14.3.11.2	Param1: wakeup_io_ctrl	180	19.4.2.1	Example	198
14.4	Functional description	180	19.5	Functional description	199
15	I/O pin configuration (IOCON)	181	19.5.1	Pin interrupts	199
			19.5.2	Pattern match engine example	199
			19.5.3	Pattern match engine edge detect examples	200

20	Secure pin interrupt and pattern match (SECPINT)	202	23.5.10.5	Miscellaneous options	231
20.1	Features	202	23.5.11	Run the SCT	231
20.2	Basic configuration	202	23.5.12	Configure the SCT without using states	231
20.2.1	Configure pins as pin interrupts or as inputs to the pattern match engine	202	23.5.13	SCT PWM example	231
20.3	Pin description	203	24	Standard counter/timers (CTIMER0 - 4)	235
20.4	General description	203	24.1	Features	235
20.4.1	Pin interrupts	203	24.2	Basic configuration	235
20.4.2	Pattern match engine	203	24.3	General description	236
20.4.2.1	Example	205	24.3.1	Capture inputs	236
20.5	Functional description	206	24.3.2	Match outputs	236
20.5.1	Pin interrupts	206	24.3.3	Applications	236
20.5.2	Pattern match engine example	206	24.3.4	Architecture	236
20.5.3	Pattern match engine edge detect examples	207	24.3.5	Peripheral input multiplexers for CTimers	237
21	Group GPIO input interrupt (GINT0)	208	24.4	Pin description	238
21.1	Features	208	24.4.1	Multiple CAP and MAT pins	238
21.2	Basic configuration	208	24.5	Functional description	238
21.3	General description	208	24.5.1	Rules for single edge controlled PWM outputs	239
21.4	Functional description	208	24.5.2	DMA operation (DMA0 and DMA1)	240
22	DMA controller (DMA0 - DMA1)	210	25	Micro-tick timer (UTICK)	241
22.1	Features	210	25.1	Features	241
22.2	Basic configuration	210	25.2	Basic configuration	241
22.3	Pin description	211	25.3	General description	241
22.4	General description	211	25.4	Pin description	242
22.4.1	DMA requests and triggers	211	26	Multirate timer (MRT)	243
22.4.1.1	DMA requests	212	26.1	Features	243
22.4.1.2	Hardware triggers	213	26.2	Basic configuration	243
22.4.1.3	Trigger operational detail	214	26.3	Pin description	243
22.4.1.4	Trigger output detail	214	26.4	General description	243
22.4.2	DMA modes	214	26.4.1	Repeat interrupt mode	244
22.4.3	Single buffer	215	26.4.2	One-shot interrupt mode	244
22.4.4	Ping-pong	216	26.4.3	One-shot stall mode	244
22.4.5	Interleaved transfers	216	27	Real-time clock (RTC)	245
22.4.6	Linked transfers (linked list)	217	27.1	Features	245
22.4.7	Address alignment for data transfer	217	27.2	Basic configuration	245
22.4.8	Channel chaining	217	27.2.1	RTC timers	246
22.4.8.1	DMA in reduced power mode	218	27.3	General description	246
23	SCTimer/PWM (SCT)	219	27.3.1	Real-time clock	246
23.1	Features	219	27.3.2	RTC subsecond counter	247
23.2	Basic configuration	220	27.3.3	High-resolution/wake-up timer	247
23.3	Pin description	221	27.3.4	General purpose backup registers	247
23.4	General description	223	27.3.5	RTC power	247
23.5	Functional description	225	27.4	Pin description	248
23.5.1	Match logic	225	28	System tick timer (SYSTICK)	249
23.5.2	Capture logic	226	28.1	Features	249
23.5.3	Event selection	226	28.2	Basic configuration	249
23.5.4	Output generation	226	28.3	General description	249
23.5.5	State logic	227	28.4	Functional description	250
23.5.6	Interrupt generation	227	28.5	Example timer calculations	250
23.5.7	Clearing the prescaler	228	29	Windowed watchdog timer (WWDT)	252
23.5.8	Match versus I/O events	228	29.1	Features	252
23.5.9	SCT operation	229	29.2	Basic configuration	252
23.5.10	Configure the SCT	229	29.3	Pin description	252
23.5.10.1	Configure the counter	229	29.4	General description	253
23.5.10.2	Configure the match and capture registers	229	29.4.1	Block diagram	253
23.5.10.3	Configure events and event responses	230	29.4.2	Clocking and power control	254
23.5.10.4	Configure multiple states	230	29.4.3	Using the WWDT lock feature	254
			29.4.3.1	Changing the WWDT reload value	255
			29.5	Functional description	255

30	Code watchdog timer (CDOG)	257	33.5.7.4	DMA as a slave receiver	277
30.1	Basic configuration	257	33.5.8	Automatic operation	277
30.2	Pin description	257	34	Universal synchronous/asynchronous receiver/transmitter (USART0 - USART2)	278
30.3	General description	257		Features	278
30.3.1	Register overview	257	34.1	Basic configuration	278
30.3.2	Operation	258	34.2	Configure the Flexcomm interface clock and USART baud rate	279
30.3.2.1	During code development debug	258	34.2.1	Configure the USART for wake-up	280
30.3.2.2	Model use cases	258	34.2.2	Wake-up from sleep mode	280
30.3.3	Faults	259	34.2.2.1	Wake-up from deep-sleep mode	280
30.3.4	Flags	259	34.2.2.2	Pin description	281
30.3.5	Counters	260	34.3	General description	283
31	OS event timer (OSTIMER)	261	34.4	Functional description	284
31.1	Features	261	34.5	AHB bus access	284
31.2	Basic configuration	261	34.5.1	Clocking and baud rates	284
31.3	Pin description	261	34.5.2	Fractional rate generator (FRG)	284
31.4	General description	261	34.5.2.1	Baud rate generator (BRG)	285
31.4.1	Central event/timestamp timer	262	34.5.2.2	32.768 kHz mode	285
31.4.2	Match, capture, and interrupt generation	262	34.5.2.3	DMA	285
31.4.2.1	Capture registers	262	34.5.3	Synchronous mode	285
31.4.2.2	Match registers and interrupt request	263	34.5.4	Flow control	285
32	Flexcomm interface serial communication (FLEXCOMM0 - 2)	264	34.5.5	Hardware flow control	285
32.1	Introduction	264	34.5.5.1	Software flow control	286
32.2	Features	264	34.5.5.2	Autobaud function	286
32.3	Basic configuration	264	34.5.6	RS-485 support	287
32.4	Architecture	264	34.5.7	Oversampling	287
32.4.1	Function summary	265	34.5.8	Break generation and detection	287
32.4.2	Choosing a peripheral function	265	34.5.9	LIN bus	287
32.4.3	FIFO usage	265	35	Serial peripheral interfaces (SPI0 - SPI2) ...	289
32.4.4	DMA	265	35.1	Features	289
32.4.5	AHB bus access	265	35.2	Basic configuration	289
32.5	Pin description	266	35.2.1	Configure the SPI for wake-up	289
33	I2C-bus interfaces (I2C0 - I2C2)	267	35.2.1.1	Wake-up from sleep mode	290
33.1	Features	267	35.2.1.2	Wake-up from deep-sleep mode	290
33.2	Pin description	267	35.3	Pin description	290
33.3	Basic configuration	267	35.4	General description	293
33.3.1	I2C transmit/receive in master mode	268	35.5	Functional description	293
33.3.1.1	Master write to slave	268	35.5.1	AHB bus access	293
33.3.1.2	Master read from slave	269	35.5.2	Operating modes: clock and phase selection	293
33.3.2	I2C receive/transmit in slave mode	270	35.5.3	Frame delays	294
33.3.2.1	Slave read from master	270	35.5.3.1	Pre_delay and Post_delay	295
33.3.2.2	Slave write to master	271	35.5.3.2	Frame_delay	295
33.3.3	Configure the I2C for wake-up	272	35.5.3.3	Transfer_delay	296
33.3.3.1	Wake-up from sleep mode	272	35.5.4	Clocking and data rates	297
33.3.3.2	Wake-up from deep-sleep mode	272	35.5.4.1	Data rate calculations	297
33.4	General description	273	35.5.5	Slave select	298
33.5	Functional description	273	35.5.6	DMA operation	298
33.5.1	AHB bus access	273	35.5.6.1	DMA master mode end-of-transfer	298
33.5.2	Bus rates and timing considerations	273	35.5.7	Data lengths greater than 16 bits	299
33.5.2.1	Rate calculations	273	35.5.8	Data stalls	299
33.5.2.2	Bus rate support	274	36	SPI flash interface (SPIFI)	301
33.5.3	Timeout	275	36.1	Features	301
33.5.4	Ten-bit addressing	275	36.2	Basic configuration	301
33.5.5	Clocking and power considerations	275	36.3	General description	301
33.5.6	Interrupt handling	276	36.4	Pin description	302
33.5.7	DMA	276	36.5	SPIFI hardware	302
33.5.7.1	DMA as a master transmitter	276			
33.5.7.2	DMA as a master receiver	276			
33.5.7.3	DMA as a slave transmitter	277			

36.6	Supported devices	302	41.2.6	Using PUF	332
36.7	Functional description	302	41.2.6.1	Order of operations	332
36.7.1	Data transfer	302	41.2.6.2	Activation code size	334
36.7.2	Software requirements and capabilities	304	41.2.6.3	Key and code sizes	334
36.7.3	Peripheral mode DMA operation	305	41.2.6.4	Key indexing	335
37	System controller (SYSCTL)	306	41.2.6.5	Key code header	335
37.1	Features	306	41.2.6.6	Key byte order on the APB interface	335
38	Programmable logical unit (PLU)	307	41.2.6.7	Enroll	336
38.1	Features	307	41.2.6.8	Start	336
38.2	Pin description	307	41.2.6.9	Generate key	336
38.3	General description	307	41.2.6.10	Set key	337
38.3.1	Using the programmable logical unit	310	41.2.6.11	Get key	337
38.3.2	Description of tool flow	311	41.2.6.12	Zeroize	337
39	CRC engine (CRC)	312	41.2.6.13	Error response	337
39.1	Features	312	41.2.6.14	Key index blocking	338
39.2	Basic configuration	312	41.2.7	Software development	338
39.3	Pin description	312	41.2.7.1	Pseudocode wait for Initialization function	339
39.4	General description	312	41.2.7.2	Pseudocode enroll function	339
39.5	Functional description	313	41.2.7.3	Pseudocode start function	339
39.5.1	CRC-CCITT setup	313	41.2.7.4	Pseudocode generate key function	340
39.5.2	CRC-16 setup	313	41.2.7.5	Pseudocode set key function	340
39.5.3	CRC-32 set-up	314	41.2.7.6	Pseudocode get key function	341
40	Trusted execution environment (TEE)	315	41.2.7.7	Pseudocode Zeroize function	342
40.1	Features	315	41.3	AES engine functional details	342
40.2	Functional description	316	41.3.1	Features	342
40.2.1	TrustZone for Armv8-M	316	41.3.2	Basic configuration	342
40.2.1.1	State transitions	318	41.3.3	General description	342
40.2.2	Attribution units	319	41.3.4	Using the AES engine	344
40.2.2.1	Device attribution unit (IDAU)	319	41.3.5	AES performance	344
40.2.2.2	Security attribution unit (SAU)	320	41.4	HASH functional details	344
40.2.2.3	Region number and test target (TT) instruction	322	41.4.1	Features	344
40.2.3	Secure AHB bus and secure AHB controller	323	41.4.2	Basic configuration	344
40.2.3.1	Memory protection checkers (MPC)	324	41.4.3	General description	345
40.2.3.2	Peripheral protection checkers (PPC)	325	41.4.4	Security lock and register access	345
40.2.3.3	Master security wrapper (MSW)	325	41.5	True random number generator (TRNG) functional details	345
40.2.3.4	Secure AHB controller	325	41.5.1	Entropy accumulation	346
40.2.4	Interrupt, DMA, and GPIO: Secure instance and masking	326	41.5.2	Initialization	346
41	Security features	327	41.5.3	Normal usage	346
41.1	Introduction	327	41.6	PRINCE real-time encryption or decryption details	346
41.1.1	Key storage/management	327	41.6.1	Functional details	346
41.1.1.1	PUF keys	328	41.6.2	Usage notes	347
41.1.2	AES engine	328	42	Cryptographic accelerator (CASPER)	348
41.1.3	SHA	328	42.1	CASPER features	348
41.1.4	Digital signatures	329	42.2	CASPER operation	348
41.1.5	Hash-based message authentication code (HMAC)	329	42.2.1	CASPER coprocessor operation	351
41.1.6	True random number generator (TRNG)	329	42.2.2	CASPER AHB operation	352
41.1.7	Universally Unique Identifier (UUID)	329	42.2.3	CASPER modes	352
41.1.8	Boot seed	330	43	Debug subsystem	354
41.1.9	PRINCE real-time encryption/decryption	330	43.1	Features	354
41.2	PUF controller and key management	330	43.2	Basic configuration	354
41.2.1	PUF controller features	330	43.3	Pin description	354
41.2.2	Basic configuration	331	43.4	Debug subsystem functional description	355
41.2.3	PUF controller operations	331	43.4.1	Debug subsystem	355
41.2.4	SRAM PUF power control	331	43.4.2	Debug access port (DAP)	356
41.2.5	Key management	331	43.4.3	CPU0 AP	356
41.2.5.1	Key loading procedure	332	43.4.4	Debugger mailbox AP	356
			43.4.5	Reset handling	356

43.4.6	Mailbox commands	358	45.1.3	OEM KEY STORE	409
43.4.6.1	Request	359	45.1.3.1	OEM KEY STORE memory map	409
43.4.6.2	Response	361	45.1.3.2	Register descriptions	409
43.4.6.3	ACK token	362	45.1.4	NXP KEY STORE	412
43.4.6.4	Error handling	363	45.1.4.1	NXP KEY STORE memory map	412
43.5	Debug session protocol	363	45.1.4.2	Register descriptions	414
43.5.1	Debug session with uninitialized/invalid flash image or ISP mode	364	45.1.5	ROM PATCH	422
43.5.2	Debug session with valid application in flash	365	45.1.5.1	ROMPATCH memory map	422
43.5.3	Debug session attaching to a running target	365	45.1.5.2	Register descriptions	422
43.5.4	Halting execution immediately following ROM execution	365	45.1.6	NMPA	423
43.6	Debug authentication	365	45.1.6.1	NMPA memory map	423
43.6.1	Debug credential certificate (DC)	367	45.1.6.2	Register descriptions	426
43.6.2	Data stored in protected flash region (PFR)	369	45.2	SRAM	446
43.6.3	Debug authentication challenge (DAC)	371	45.3	External flash	447
43.6.4	Debug authentication response (DAR)	372	45.4	APB peripherals	448
43.6.5	Debug port states before entering user code	374	45.4.1	System control (SYSCON)	449
43.6.6	Fault analysis (FA) mode	375	45.4.1.1	SYSCON memory map	449
43.6.7	Device processing the DAR	375	45.4.1.2	Register descriptions	453
43.6.7.1	Successful authentication	375	45.4.2	I/O pin configuration (IOCON)	511
43.6.7.2	Debug port states before entering user code	376	45.4.2.1	IOCON memory map	511
43.6.8	Glossary	376	45.4.2.2	Register descriptions	511
44	2.4 GHz radio (Bluetooth Low Energy)	378	45.4.3	Group GPIO input interrupt (GINT)	514
44.1	Features	378	45.4.3.1	GINT memory map	514
44.2	Bluetooth Low Energy link layer	378	45.4.3.2	Register descriptions	514
44.2.1	Link Layer HCI	379	45.4.4	Pin interrupt and pattern match (PINT)	516
44.2.2	HCI vendor-specific commands	379	45.4.4.1	PINT memory map	516
44.2.2.1	HCI LE set custom event mask	379	45.4.4.2	Register descriptions	517
44.2.2.2	HCI release version	380	45.4.5	Input multiplexing (INPUT MUX)	530
44.2.2.3	HCI set PHY TX power	380	45.4.5.1	INPUTMUX memory map	530
44.2.2.4	HCI set BD address	380	45.4.5.2	Register descriptions	532
44.2.2.5	HCI periodic PHY calibration	381	45.4.6	Standard counter/timers (CTIMER0 to 4)	550
44.2.2.6	HCI set deep sleep mode enable	381	45.4.6.1	CTIMER memory map	550
44.2.2.7	HCI get remaining time for next event	381	45.4.6.2	Register descriptions	551
44.2.2.8	HCI set event setup time value	382	45.4.7	Windowed watchdog timer (WWDT)	562
44.2.2.9	HCI select sleep clock source	382	45.4.7.1	WWDT memory map	562
44.2.2.10	HCI set connection TX power command	383	45.4.7.2	Register descriptions	563
44.2.2.11	HCI set RSSI golden range command	384	45.4.8	Multi-rate timer (MRT)	566
44.2.2.12	HCI select TX power mode command	384	45.4.8.1	MRT memory map	566
44.2.2.13	HCI curb sleep state	385	45.4.8.2	Register descriptions	567
44.2.2.14	HCI locally terminate ACL connection	386	45.4.9	Micro-tick timer (UTICK)	572
44.2.3	HCI external custom commands	386	45.4.9.1	UTICK memory map	572
44.2.4	Bluetooth Low Energy power management ...	387	45.4.9.2	Register descriptions	572
44.2.5	TX power control	388	45.4.10	Analog control (ANALOG_CTRL)	576
44.2.6	Continuous wave	390	45.4.10.1	ANACTRL memory map	576
44.3	Bluetooth Low Energy host	391	45.4.10.2	Register descriptions	577
45	Register description	391	45.4.11	Power management control (PMC)	586
45.1	Flash	392	45.4.11.1	PMC memory map	586
45.1.1	Customer field programmable area (CFPA) ...	392	45.4.11.2	Register descriptions	589
45.1.1.1	CFPA memory map	392	45.4.12	System configuration (SYSCTL)	623
45.1.1.2	Register descriptions	393	45.4.12.1	SYSCTL memory map	623
45.1.2	Customer manufacturing programmable area (CMPA)	399	45.4.12.2	Register descriptions	624
45.1.2.1	CMPA memory map	399	45.4.13	Real-time clock (RTC)	627
45.1.2.2	Register descriptions	400	45.4.13.1	RTC memory map	627
			45.4.13.2	Register descriptions	627
			45.4.14	Synchronous OS/event timer with wake-up timer	632
			45.4.14.1	OSTIMER memory map	632
			45.4.14.2	Register descriptions	632
			45.4.15	Flash memory unit (FMU)	635

45.4.15.1	FMU memory map	635	45.5.12.1	HASHAES memory map	814
45.4.15.2	Register descriptions	636	45.5.12.2	Register descriptions	815
45.4.16	NVM PRINCE encryption and decryption (NPX)	646	45.5.13	Crypto-coprocessor (CASPER)	825
45.4.16.1	NPX memory map	646	45.5.13.1	CASPER memory map	825
45.4.16.2	Register descriptions	646	45.5.13.2	Register descriptions	826
45.4.17	True random number generator (TRNG)	650	45.5.14	AHB security control (AHB_SECURE_CTRL)	832
45.4.17.1	TRNG memory map	650	45.5.14.1	Security control memory map	832
45.4.17.2	Register descriptions	652	45.5.14.2	Register descriptions	834
45.4.18	Physical unclonable function control (PUF) ...	675	45.5.15	Bluetooth Low Energy	885
45.4.18.1	PUF memory map	675	45.6	Arm NVIC and TrustZone	886
45.4.18.2	Register descriptions	676	45.6.1	Nested vector interrupt control (NVIC)	886
45.4.19	PUF SRAM control	687	45.6.1.1	NVIC memory map	886
45.4.19.1	PUF SRAM control memory map	687	45.6.1.2	Register descriptions	887
45.4.19.2	Register descriptions	687	45.6.2	Arm system control block registers (SCB)	890
45.4.20	Programmable logic unit (PLU)	692	45.6.2.1	SCB memory map	890
45.4.20.1	PLU memory map	692	45.6.2.2	Register descriptions	891
45.4.20.2	Register descriptions	693	45.6.3	Security attribution unit (SAU)	898
45.4.21	ROM control and patch (ROMCP)	698	45.6.3.1	SAU memory map	898
45.4.21.1	ROMCP memory map	698	45.6.3.2	Register descriptions	899
45.4.21.2	Register descriptions	698	46	References	904
45.5	AHB peripherals	703	47	Abbreviations	905
45.5.1	DMA engine (DMA0)	704	48	Revision history	907
45.5.1.1	DMA memory map	704		Legal information	908
45.5.1.2	Register descriptions	705			
45.5.2	State configurable timer/PWM (SCT)	717			
45.5.2.1	SCT memory map	717			
45.5.2.2	Register descriptions	718			
45.5.3	Flexcomm interface	736			
45.5.3.1	Flexcomm memory map	736			
45.5.3.2	Register descriptions	736			
45.5.4	USART	738			
45.5.4.1	USART memory map	738			
45.5.4.2	Register descriptions	739			
45.5.5	Serial peripheral interfaces (SPI)	756			
45.5.5.1	SPI memory map	756			
45.5.5.2	Register descriptions	757			
45.5.6	I2C-bus interfaces	769			
45.5.6.1	I2C memory map	769			
45.5.6.2	Register descriptions	770			
45.5.7	General-purpose I/O	790			
45.5.7.1	GPIO memory map	790			
45.5.7.2	Register descriptions	791			
45.5.8	CRC engine (CRC)	795			
45.5.8.1	CRC memory map	795			
45.5.8.2	Register descriptions	795			
45.5.9	Debug mailbox (DBG)	797			
45.5.9.1	MCU debugger mailbox memory map	797			
45.5.9.2	Register descriptions	797			
45.5.10	SPIFI slave port (SPIFI)	799			
45.5.10.1	SPIFI memory map	799			
45.5.10.2	Register descriptions	800			
45.5.11	Code watchdog (CDOG)	806			
45.5.11.1	CDOG memory map	806			
45.5.11.2	Register descriptions	807			
45.5.12	Hash-AES peripheral (HASHAES)	814			

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.